Developer Tools State of the Union

Session 400

These are confidential sessions—please refrain from streaming, blogging, or taking pictures.

At the convention center, a movie will be placed here

Developer Tools State of the Union

Andreas Wendker
Sr. Director, Development Technologies

Finder File Edit View Go Window Help
Mon 10:50 AM

3G
9:42 AM
Text
Tuesday
9
Calendar
Photos
Camera
YouTube
Stocks
280
Maps
73°
Weather
Clock
Calculator
Notes
Settings
iTunes
App Store
Phone
Mail
Safari
iPod

1+ Million Downloads

Mac OS X
Snow Leopard

Snow Leopard Preview

iPhone SDK 3.0

DVD

Download

Demo

Building an iPhone application with Xcode and Interface Builder

Matthew Firlik
Sr. Manager, Cocoa Development Technologies

Color palette
Liberation Library & Fonts
Preview component error fonts, font styles, & pairings open settings
Spacing tools
Font spacing
Alignment
Color tool
working space
Comparison viewing
PROJECT: APPLICATION .AP
VIEW
ARCHITECT: XCODE

no. 2 graphite pencil with eraser
brush
ruler

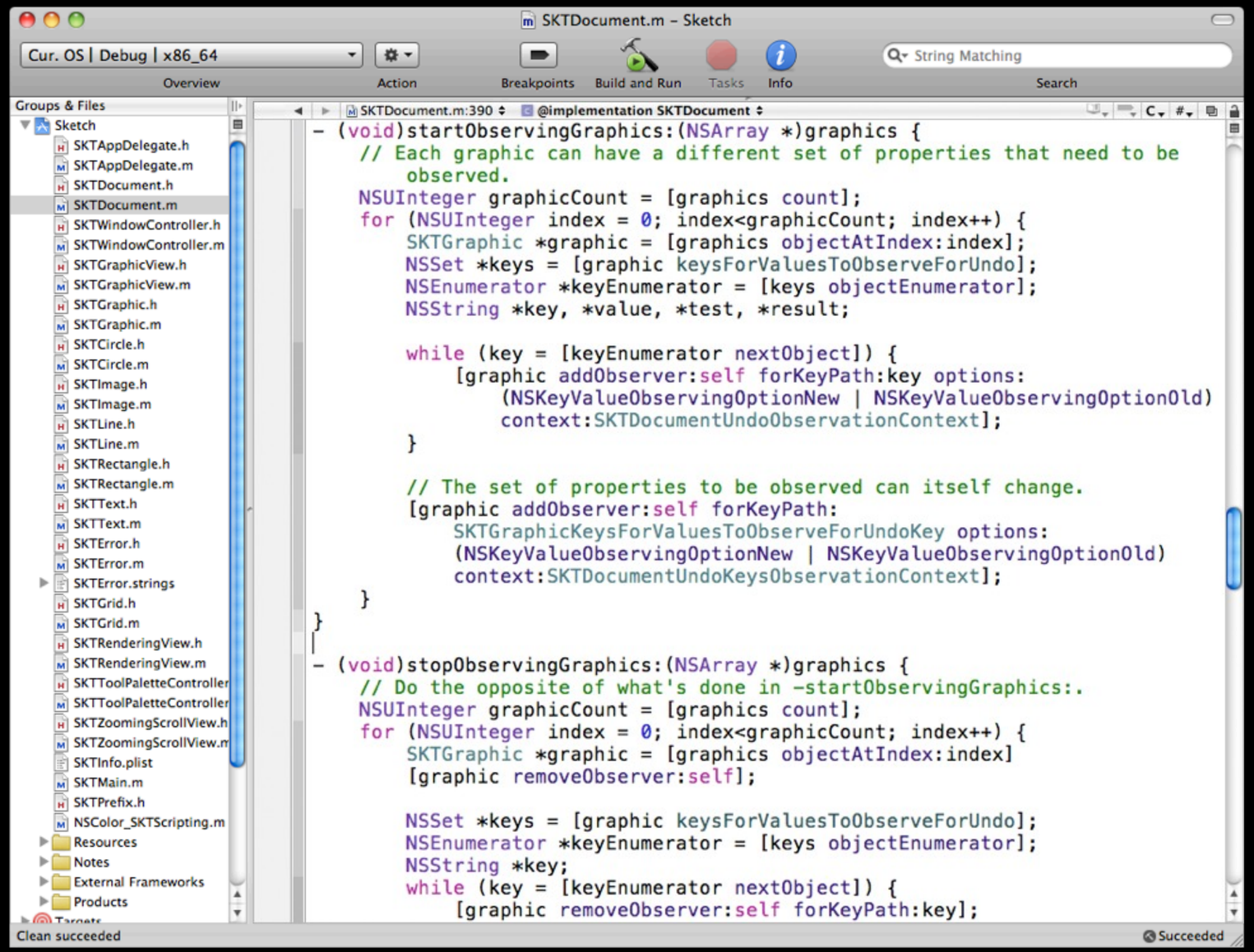
Many, Many Improvements

Reacted to your feedback

- Xcode
  - Project Find performance up to 2.5x faster
  - Code completion fidelity
  - Project renaming
- Interface Builder
  - Copy & Paste connections and bindings
  - Edit classes in Library
  - Support for new iPhone classes and functionalities
    - Map Kit
    - Accessibility

Xcode Source Editor

Source code at center stage



Xcode Source Editor

Source code at center stage

SKTDocument.m:390 @implementation SKTDocument

```
- (void)startObservingGraphics:(NSArray *)graphics {
    // Each graphic can have a different set of properties that need to be
        observed.
    NSUInteger graphicCount = [graphics count];
    for (NSUInteger index = 0; index<graphicCount; index++) {
        SKTGraphic *graphic = [graphics objectAtIndex:index];
        NSSet *keys = [graphic keysForValuesToObserveForUndo];
        NSEnumerator *keyEnumerator = [keys objectEnumerator];
        NSString *key, *value, *test, *result;

        while (key = [keyEnumerator nextObject]) {
            [graphic addObserver:self forKeyPath:key options:
                (NSKeyValueObservingOptionNew | NSKeyValueObservingOptionOld)
                context:SKTDocumentUndoObservationContext];
        }

        // The set of properties to be observed can itself change.
        [graphic addObserver:self forKeyPath:
            SKTGraphicKeysForValuesToObserveForUndoKey options:
            (NSKeyValueObservingOptionNew | NSKeyValueObservingOptionOld)
            context:SKTDocumentUndoKeysObservationContext];
    }
}

- (void)stopObservingGraphics:(NSArray *)graphics {
    // Do the opposite of what's done in -startObservingGraphics:.
    NSUInteger graphicCount = [graphics count];
    for (NSUInteger index = 0; index<graphicCount; index++) {
        SKTGraphic *graphic = [graphics objectAtIndex:index]
        [graphic removeObserver:self];

        NSSet *keys = [graphic keysForValuesToObserveForUndo];
        NSEnumerator *keyEnumerator = [keys objectEnumerator];
        NSString *key;
        while (key = [keyEnumerator nextObject]) {
            [graphic removeObserver:self forKeyPath:key];
```

Xcode Source Editor

Source code at center stage

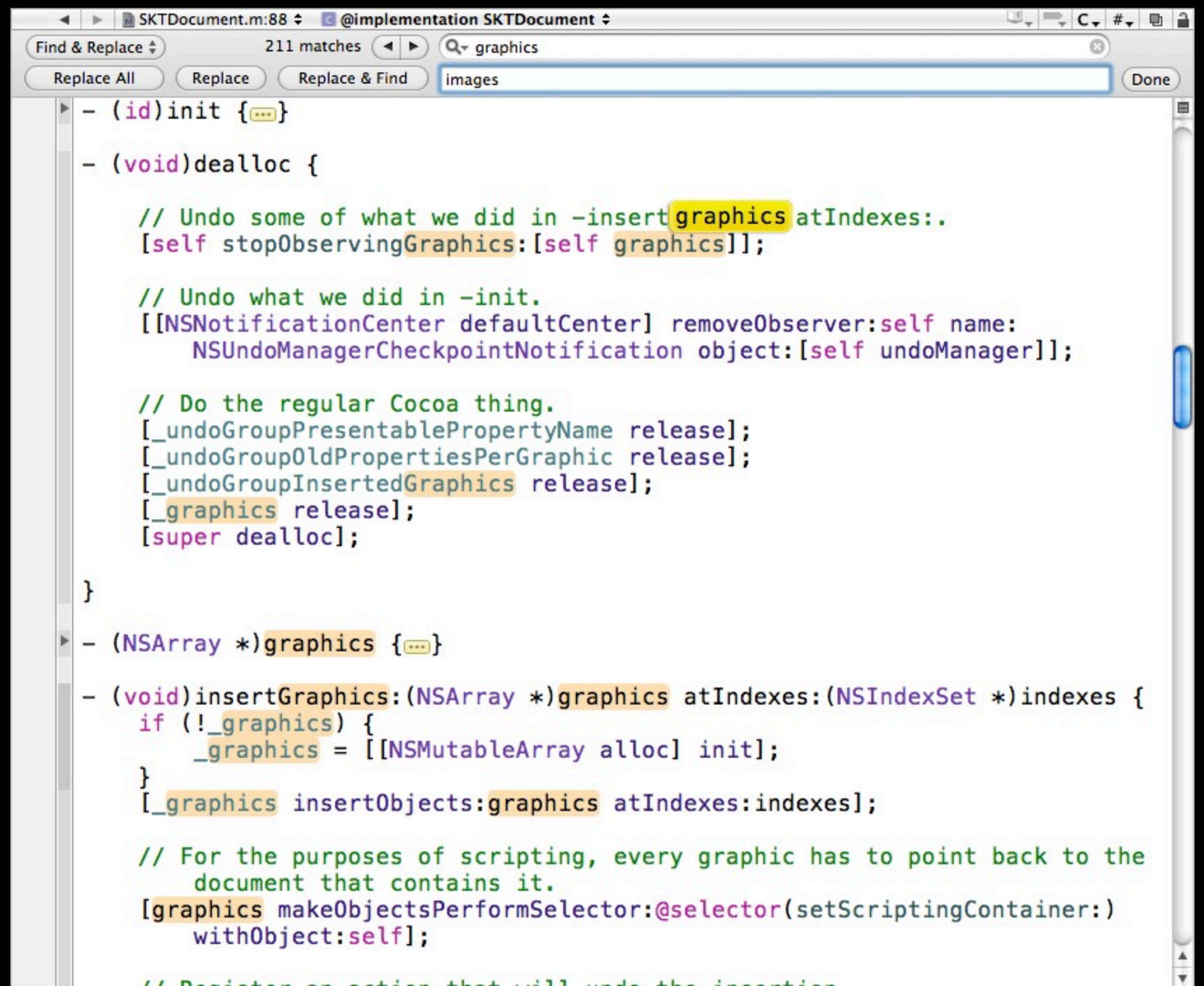
SKTDocument.m:390 @implementation SKTDocument

```
- (void)startObservingGraphics:(NSArray *)graphics {
    // Each graphic can have a different set of properties that need to be
        observed.
    NSUInteger graphicCount = [graphics count];
    for (NSUInteger index = 0; index<graphicCount; index++) {
        SKTGraphic *graphic = [graphics objectAtIndex:index];
        NSSet *keys = [graphic keysForValuesToObserveForUndo];
        NSEnumerator *keyEnumerator = [keys objectEnumerator];
        NSString *key, *value, *test, *result;

        while (key = [keyEnumerator nextObject]) {
            [graphic addObserver:self forKeyPath:key options:
                (NSKeyValueObservingOptionNew | NSKeyValueObservingOptionOld)
                context:SKTDocumentUndoObservationContext];
        }

        // The set of properties to be observed can itself change.
        [graphic addObserver:self forKeyPath:
            SKTGraphicKeysForValuesToObserveForUndoKey options:
            (NSKeyValueObservingOptionNew | NSKeyValueObservingOptionOld)
            context:SKTDocumentUndoKeysObservationContext];
    }
}

- (void)stopObservingGraphics:(NSArray *)graphics {
    // Do the opposite of what's done in -startObservingGraphics:.
    NSUInteger graphicCount = [graphics count];
    for (NSUInteger index = 0; index<graphicCount; index++) {
        SKTGraphic *graphic = [graphics objectAtIndex:index]
        [graphic removeObserver:self];

        NSSet *keys = [graphic keysForValuesToObserveForUndo];
        NSEnumerator *keyEnumerator = [keys objectEnumerator];
        NSString *key;
        while (key = [keyEnumerator nextObject]) {
            [graphic removeObserver:self forKeyPath:key];
```

- Code completion
- Code folding
- Syntax coloring
- Customizable
- New coding font Menlo
- Debugger bar and data tips

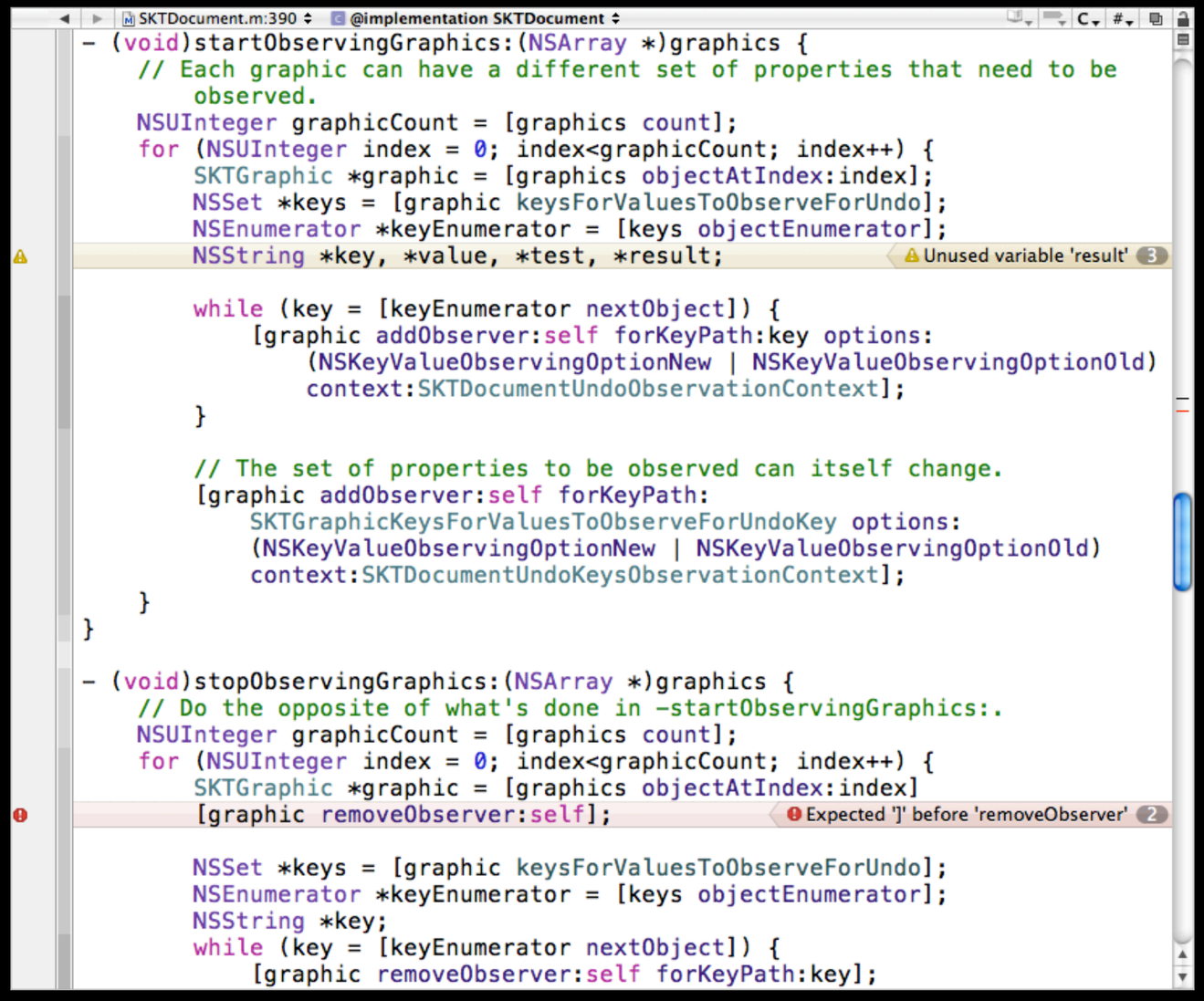
Xcode Source Editor

Integrated Find & Replace



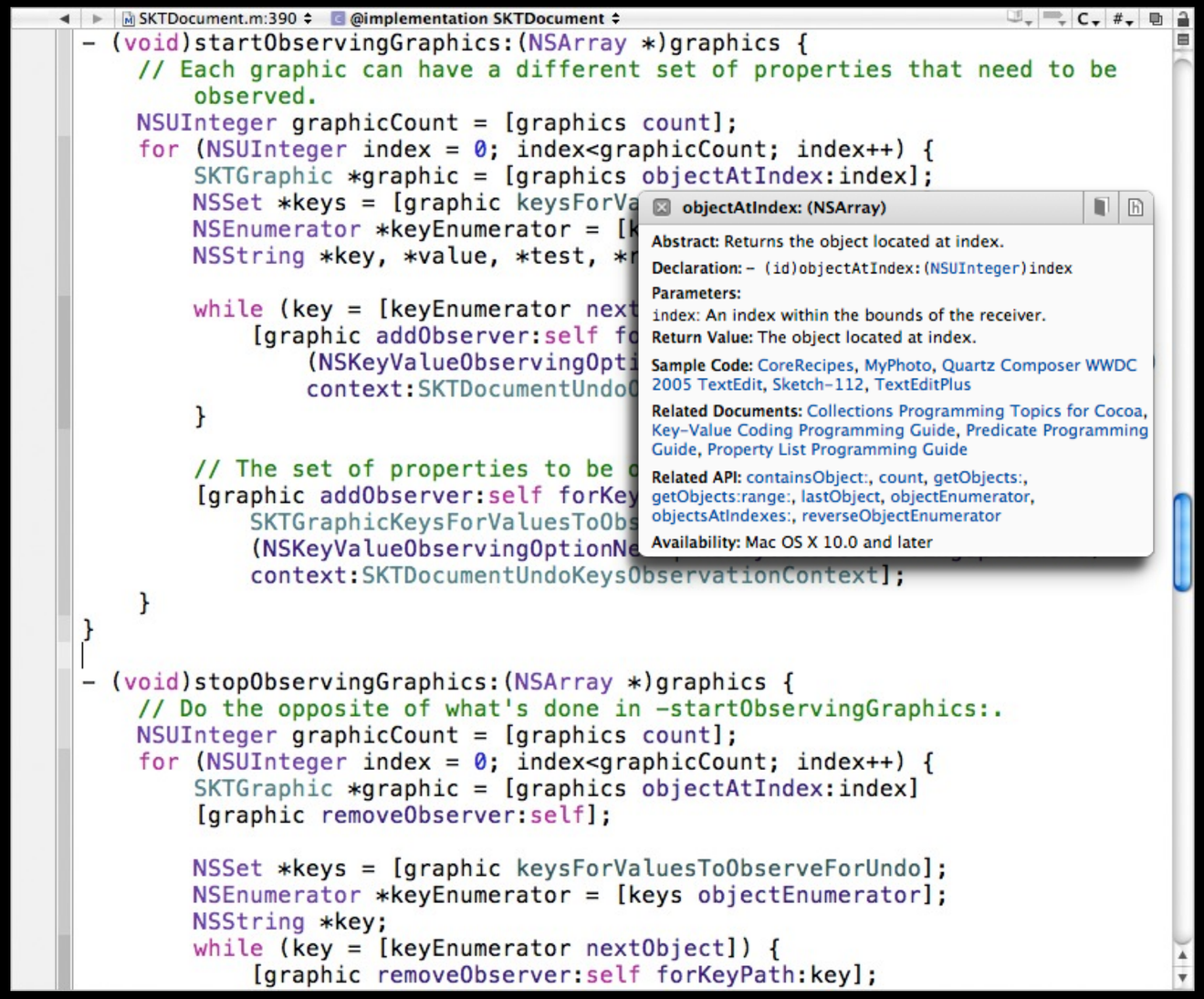
Xcode Source Editor

Improved message bubbles



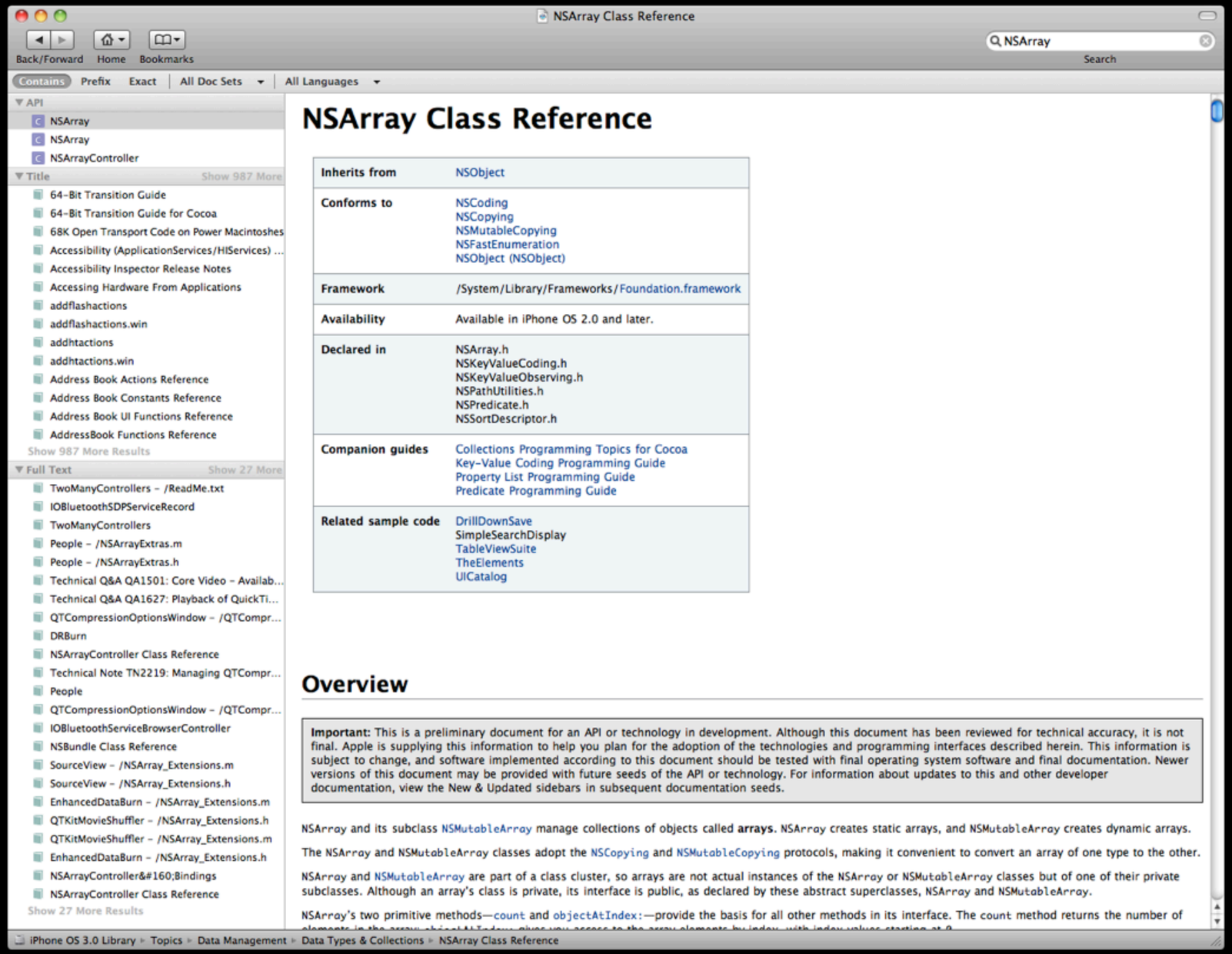
Xcode Source Editor

Quick Help



Documentation Window

Focus on content



Color palette
PROJECT: APPLICATION
XCODE

no. 2 graphite pencil with eraser
brush
ruler

</SCRIPT>
DOCUMENT 1/2
DASHBOARD WIDGET
DRAFT RS2
APPLE COMPUTER, INC.

Dashcode

Widgets

Dashcode

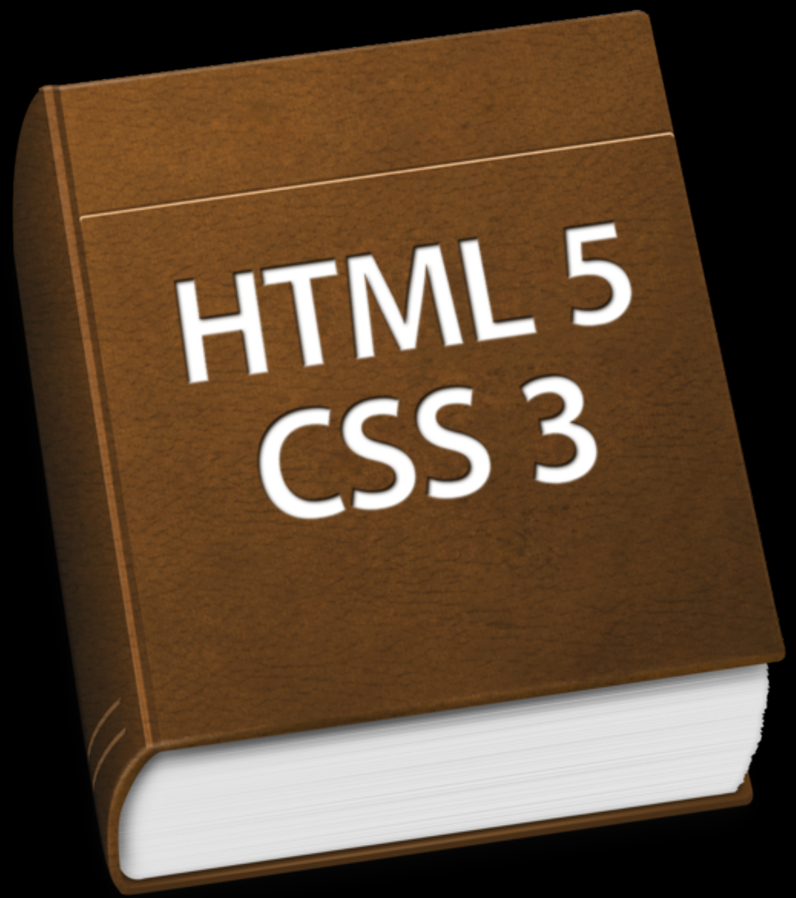
Web applications for Mobile Safari

Dashcode

Web applications for Safari

Dashcode

New standards



- CSS animations and effects
- Media tags
- Offline storage

Dashcode

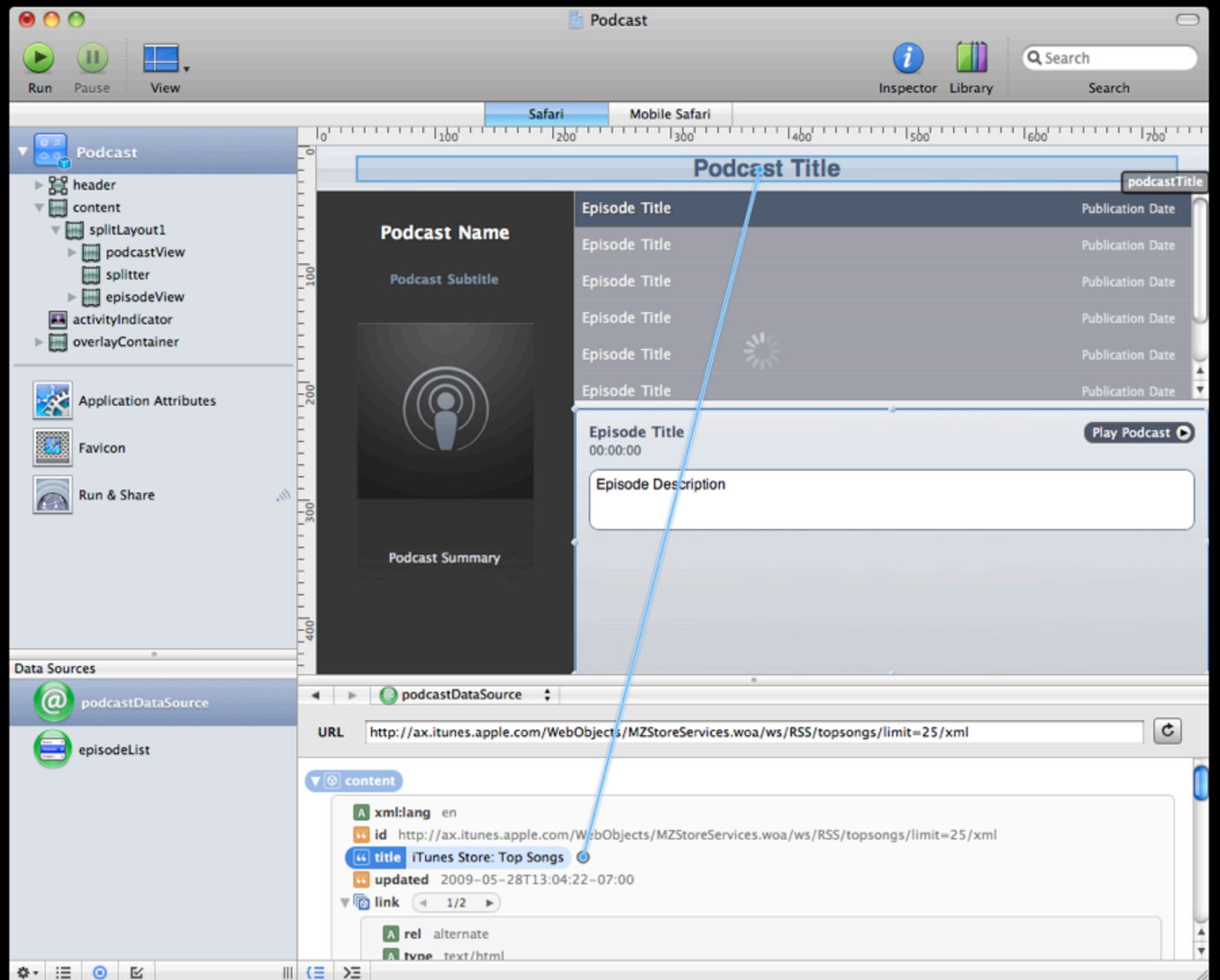
Data sources and bindings

Dashcode

Data sources and bindings

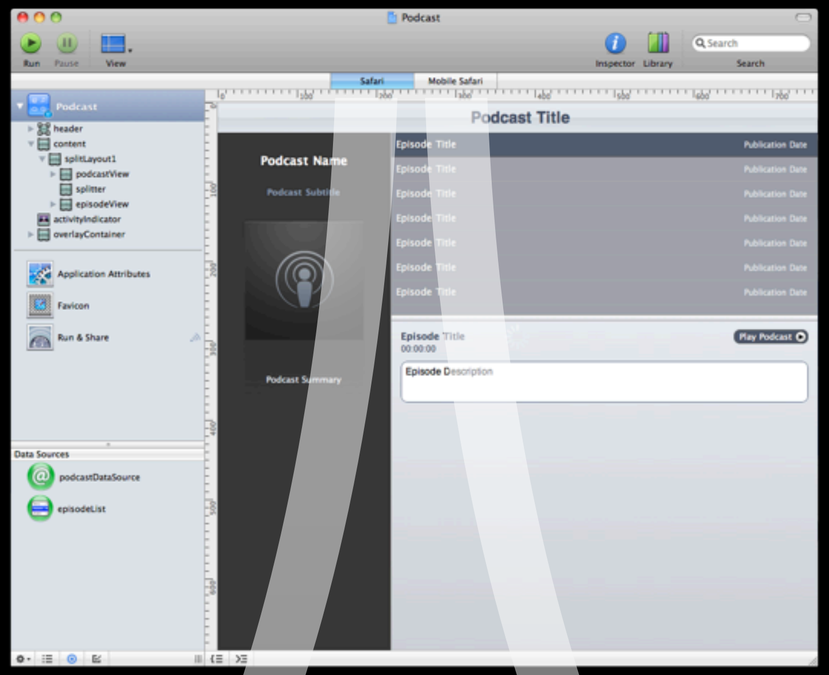
Dashcode

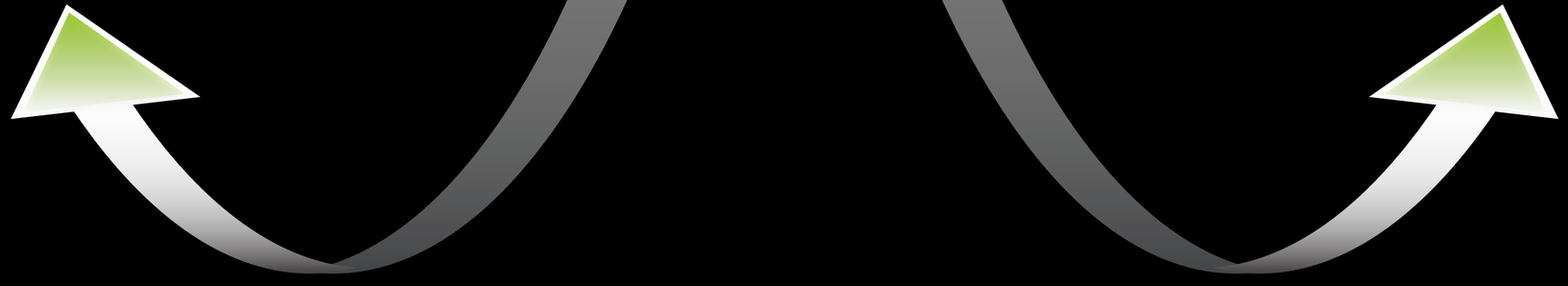
Data sources and bindings



Dashcode

Multiple deployment targets





Demo

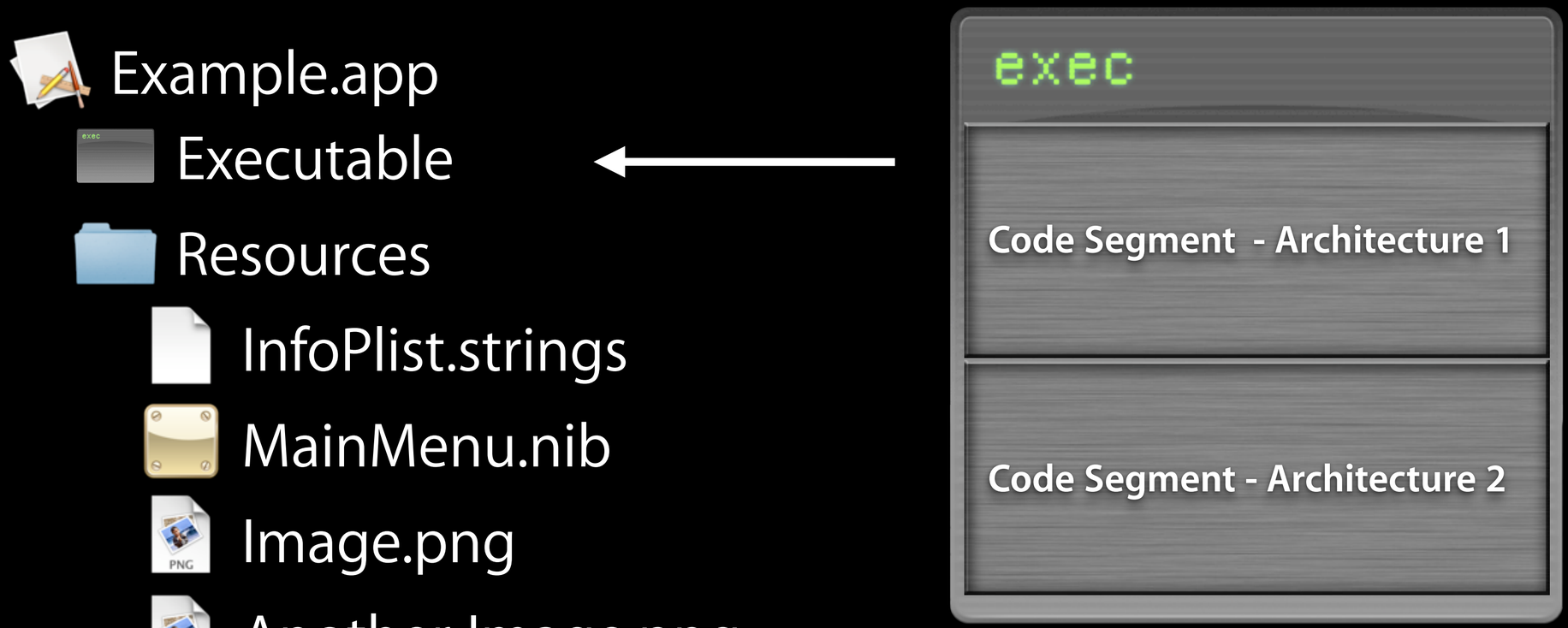
Building a web application with Dashcode

Max Drukman
Sr. Human Interface Designer

Getting the Most Out of Xcode Tools

Multiple Processor Architectures

Xcode builds and merges multiple binaries



No Worries

Distribute single application

Install on all systems

OS picks best match

Mac OS X

Universal applications

- Run on all Intel and PowerPC Macs
- For Snow Leopard, start including Intel 64-bit

iPhone OS

Optimized applications

iPhone
and
iPhone 3G

iPhone 3G S

ARM v6

ARM v6 and ARM v7

Compilers

New default compiler

- GCC 4.2 is new default compiler
  - Snow Leopard (support for Blocks)
  - iPhone OS 3.0 (support for ARM v7 code generation)
- Stronger optimizations and autovectorization

LLVM

Clang Parser for Objective-C & C

LLVM Code Generator

LLVM

Chris Lattner
Sr. Manager, Compilers and Low-Level Tools

Why a New Compiler Technology?

- Affects entire system
- Diminishing returns with GCC
- Time for a revolution

Apple's New Compiler Technology

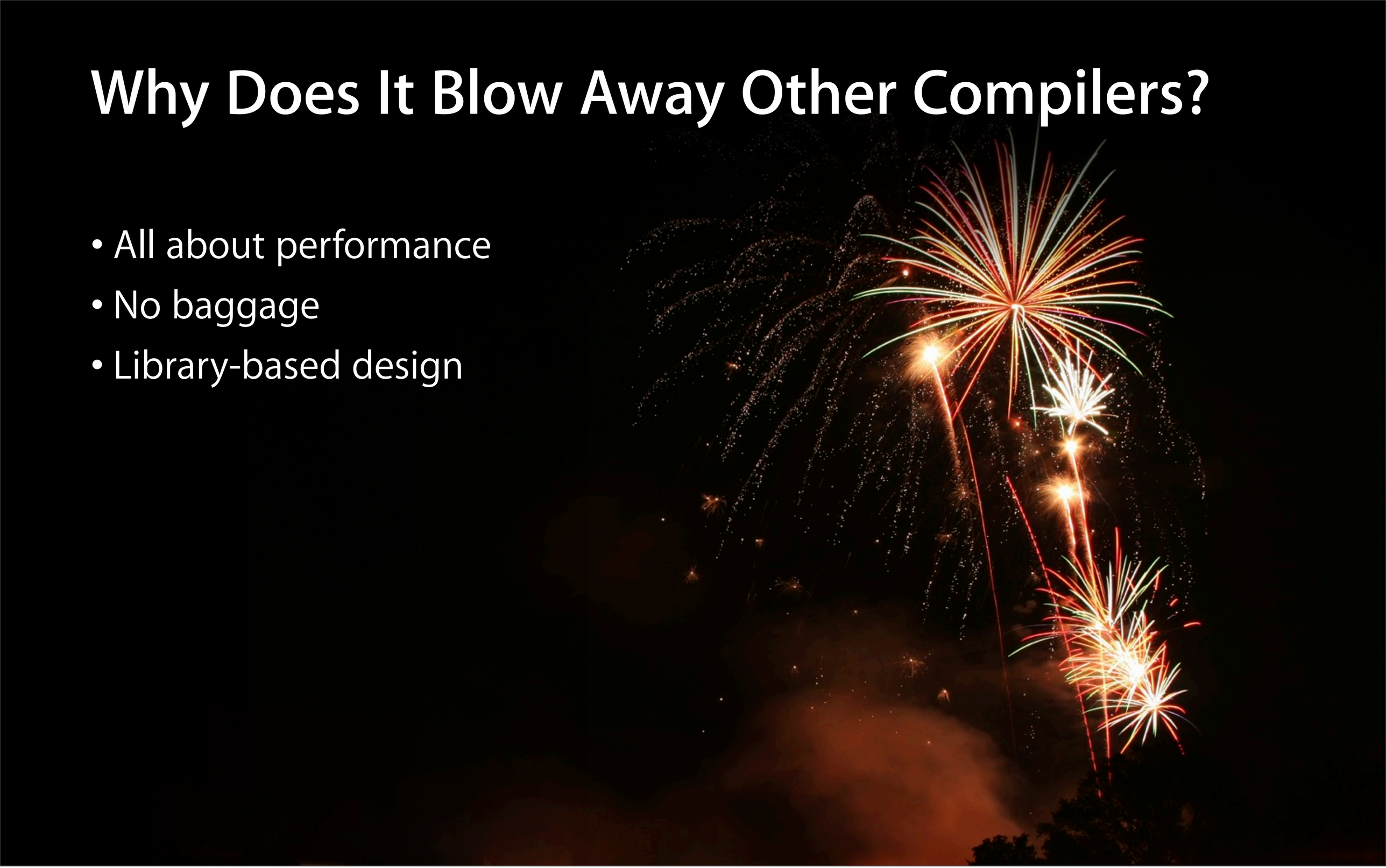
- Open Source project

http://llvm.org/

- Apple contributing for almost 4 years

Why Does It Blow Away Other Compilers?

- All about performance
- No baggage
- Library-based design

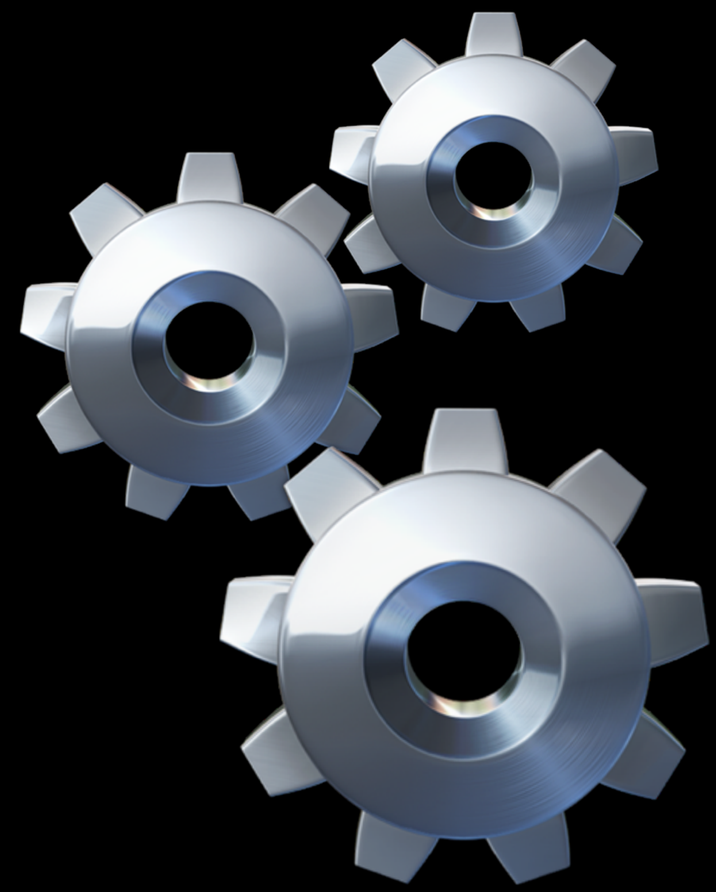


LLVM Library Design Goals

- Modular
- Small footprint
- Language independent
- Target many architectures

LLVM Library Design Goals

- Modular
- Small footprint
- Language independent
- Target many architectures

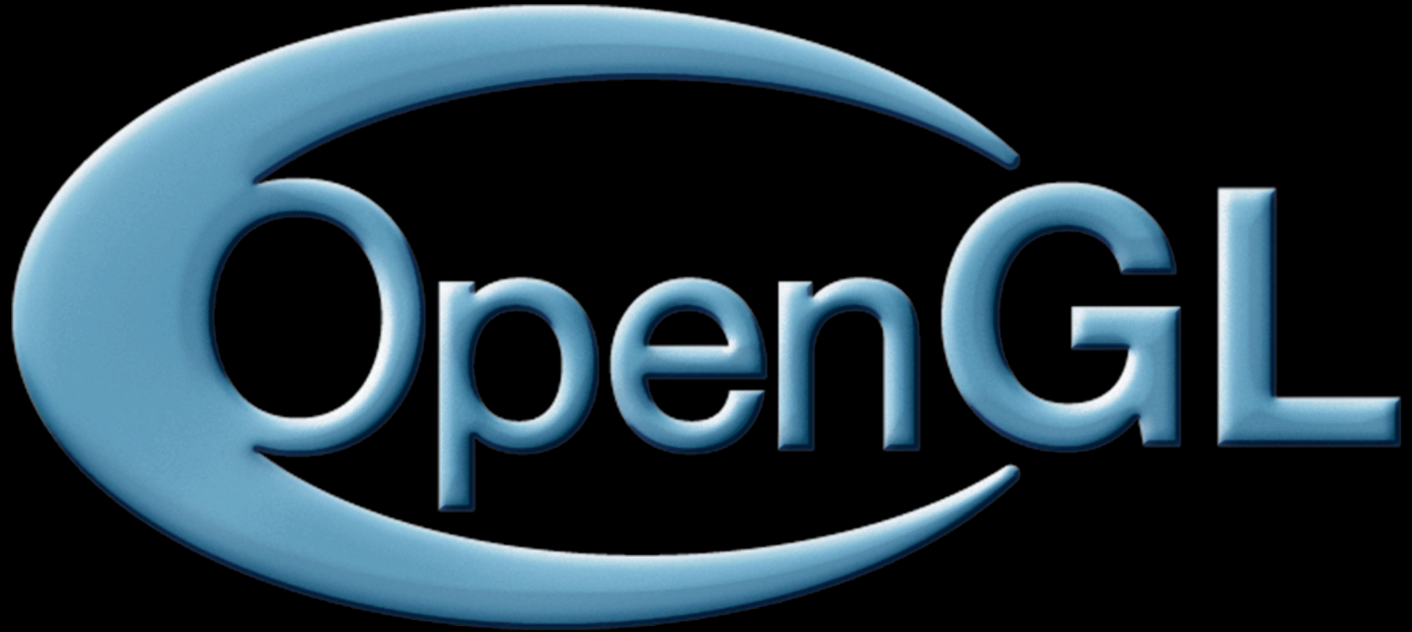


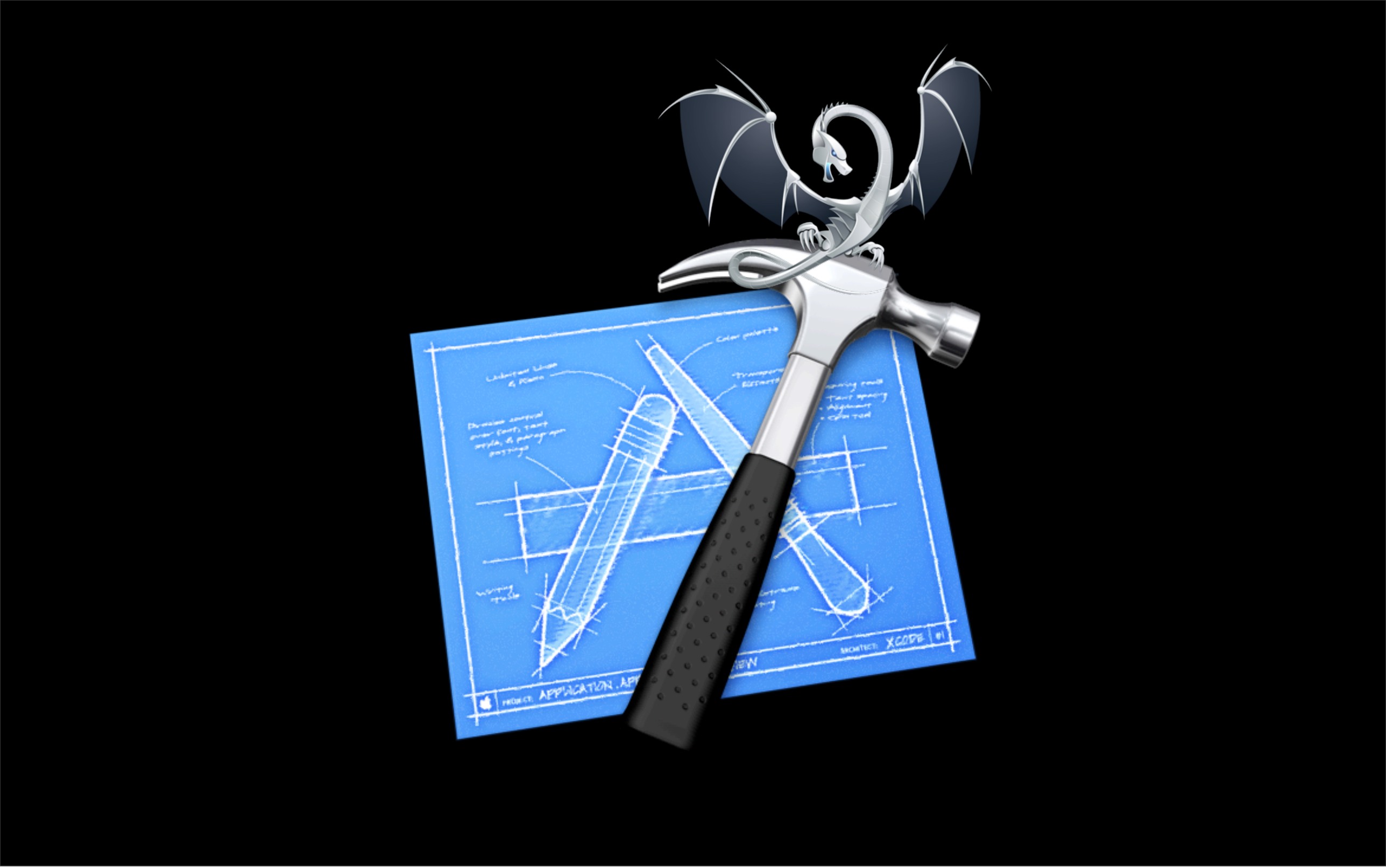
OpenCL

- Compile and optimize code
  ... at runtime in your app
  ... for CPUs *and* GPUs
  ... that is fully general C
  ... when performance is everything!

OpenGL

- Runtime optimize algorithms in OpenGL
- Many advantages over previous JIT

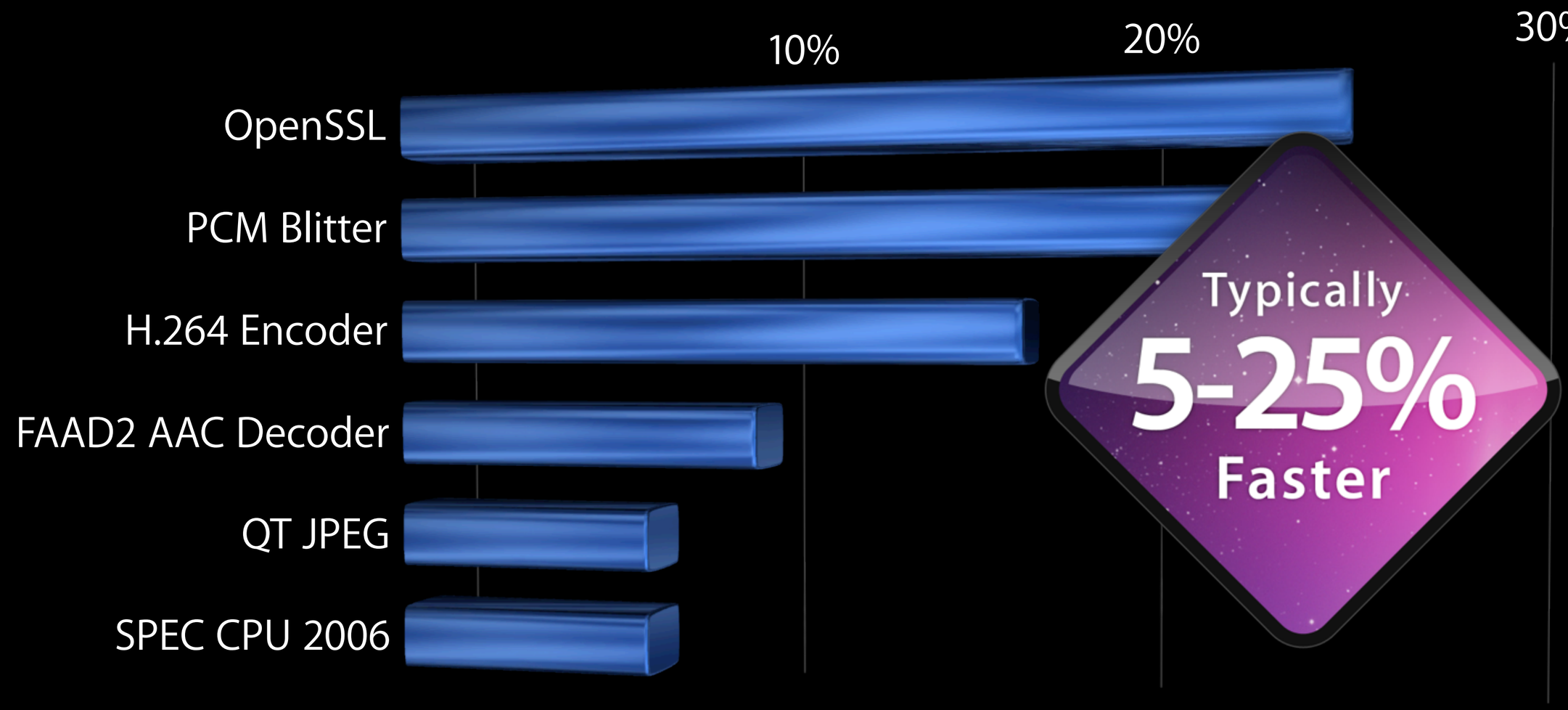



Color palette
XCODE

LLVM and Clang Score Card

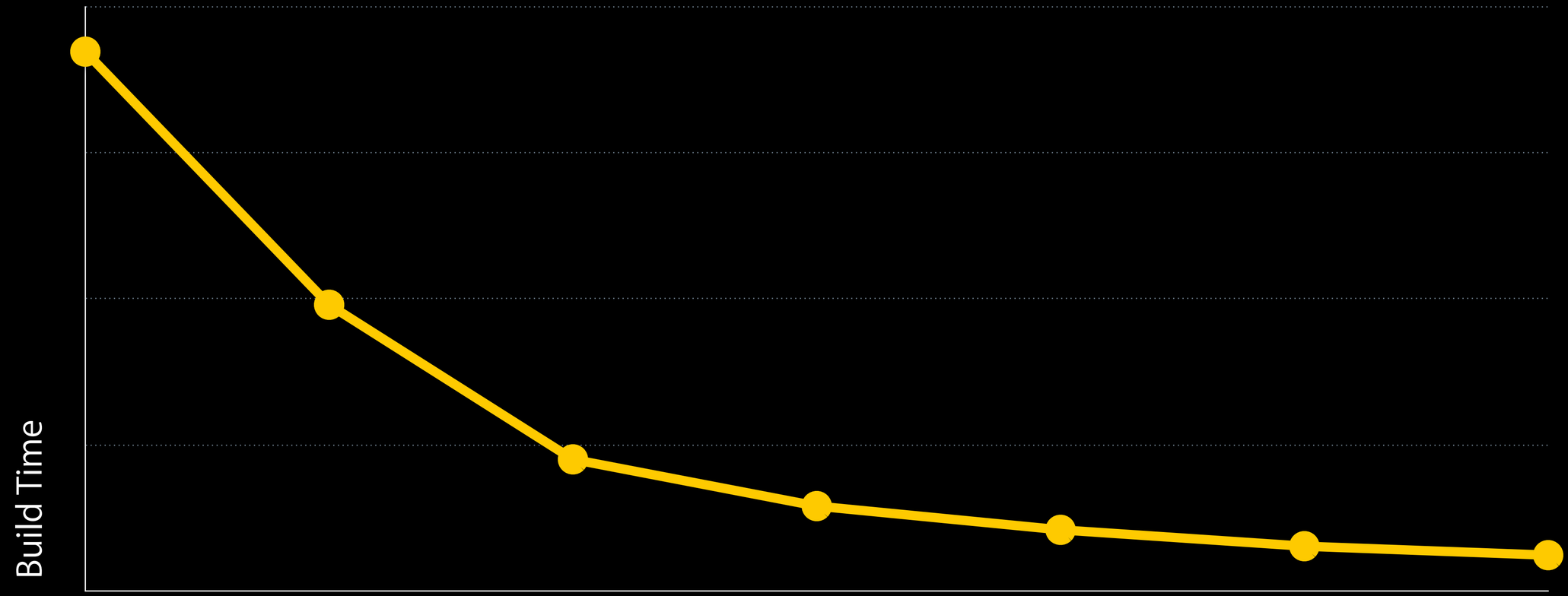
Speed of Generated Code

LLVM vs. GCC code generator



LLVM and Clang Score Card

Build Times



Panther	**Tiger**	**Tiger**	**Tiger**	**Tiger**	**Leopard**	**Snow Leopard**
Xcode 1.0	Xcode 2.0	Xcode 2.4	Xcode 2.4	Xcode 2.4	Xcode 3.0	Xcode 3.1
PPC G4	PPC G5	PPC G5	Intel	Intel	Intel	Intel
2 Cores	2 Cores	4 Cores	4 Cores	4 Cores	4 Cores	8 Cores

2003 **2008**

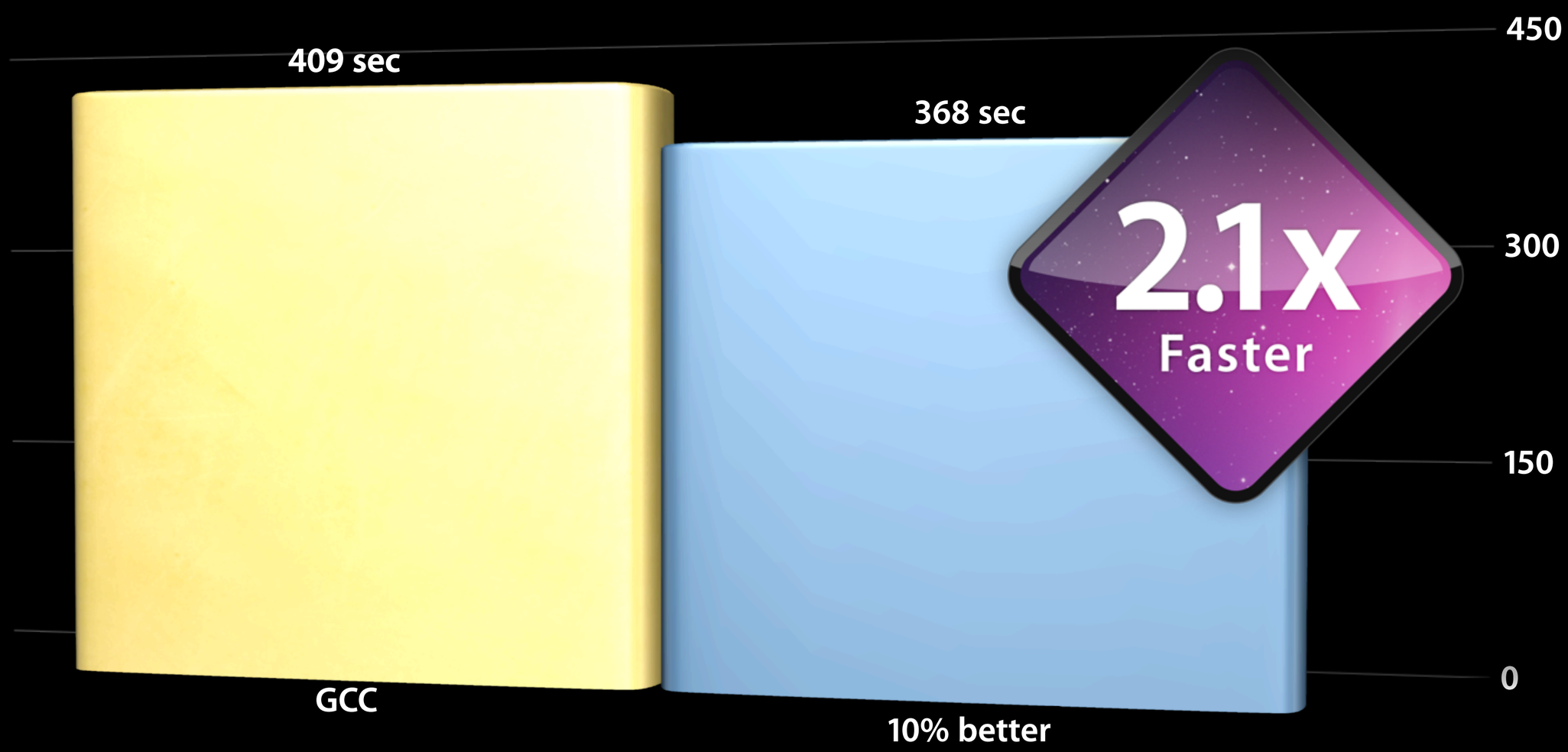
Build Times

Complete project build time: Xcode

8 cores, 16 threads

Build Times

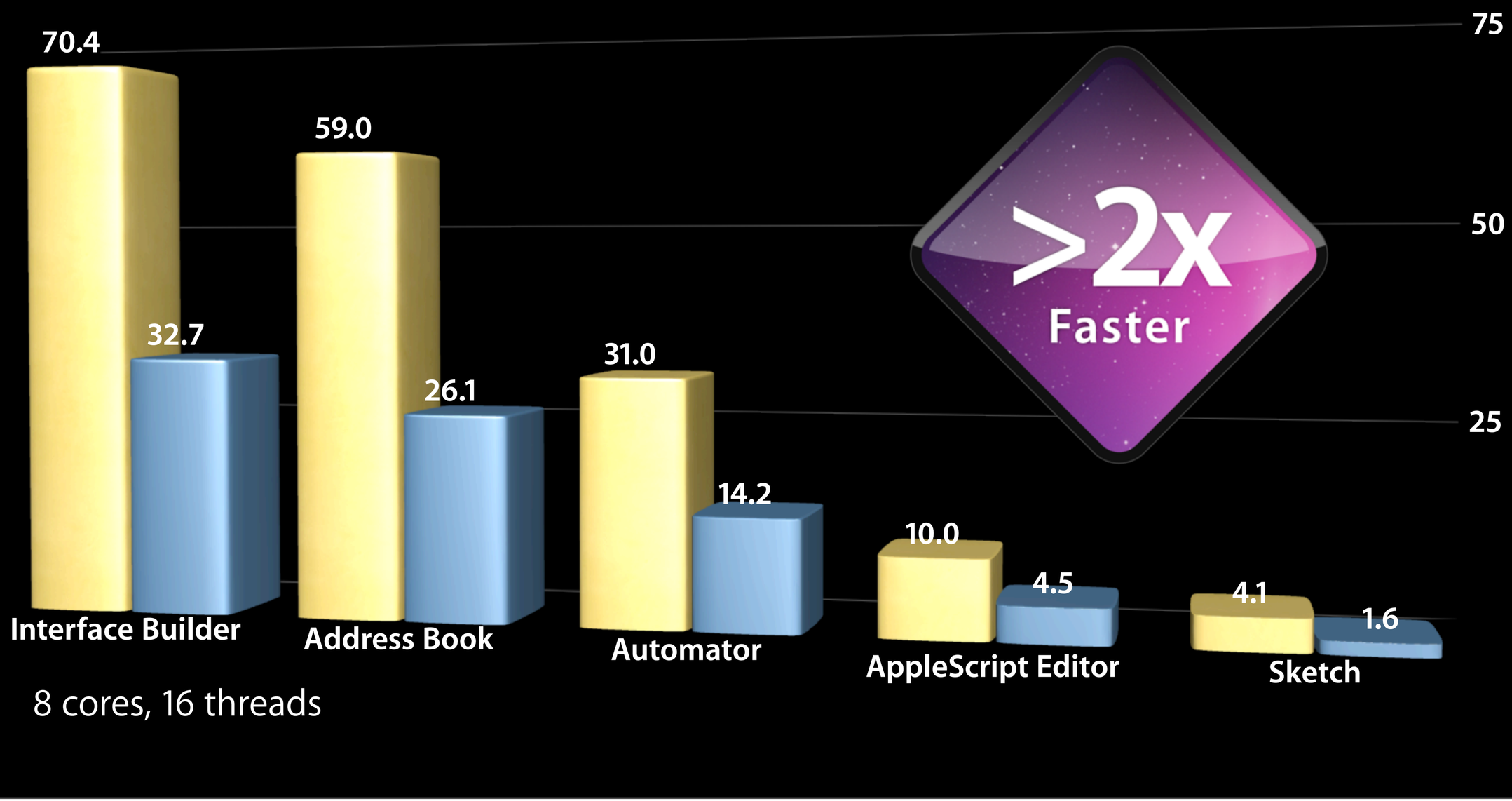
Complete project build time: Xcode



8 cores, 16 threads

Build Times

Clang vs. GCC complete project build times



8 cores, 16 threads

LLVM and Clang Score Card

Diagnostics

Senseless piece of code

```
struct A { int F; } Str;

int compute1(int, float, struct A, int, double, double);
int compute2(int);

int foo(struct A *Ptr, int X) {
    compute1(X, 4.0f, X + 42, 92, 3.0f, 2.0f);

    return X + compute2(X ? ((Str.F + 40) + Str) / 42 + Str.F : Ptr->F);
}
```

Diagnostics

GCC error messages in Xcode

```
struct A { int F; } Str;

int compute1(int, float, struct A, int, double, double);
int compute2(int);

int foo(struct A *Ptr, int X) {
    compute1(X, 4.0f, X + 42, 92, 3.0f, 2.0f);          Incompatible type for argument 3 of 'compute1'

    return X + compute2(X ? ((Str.F + 40) + Str) / 42 + Str.F : Ptr->F);          Invalid operands to binary +
}
```

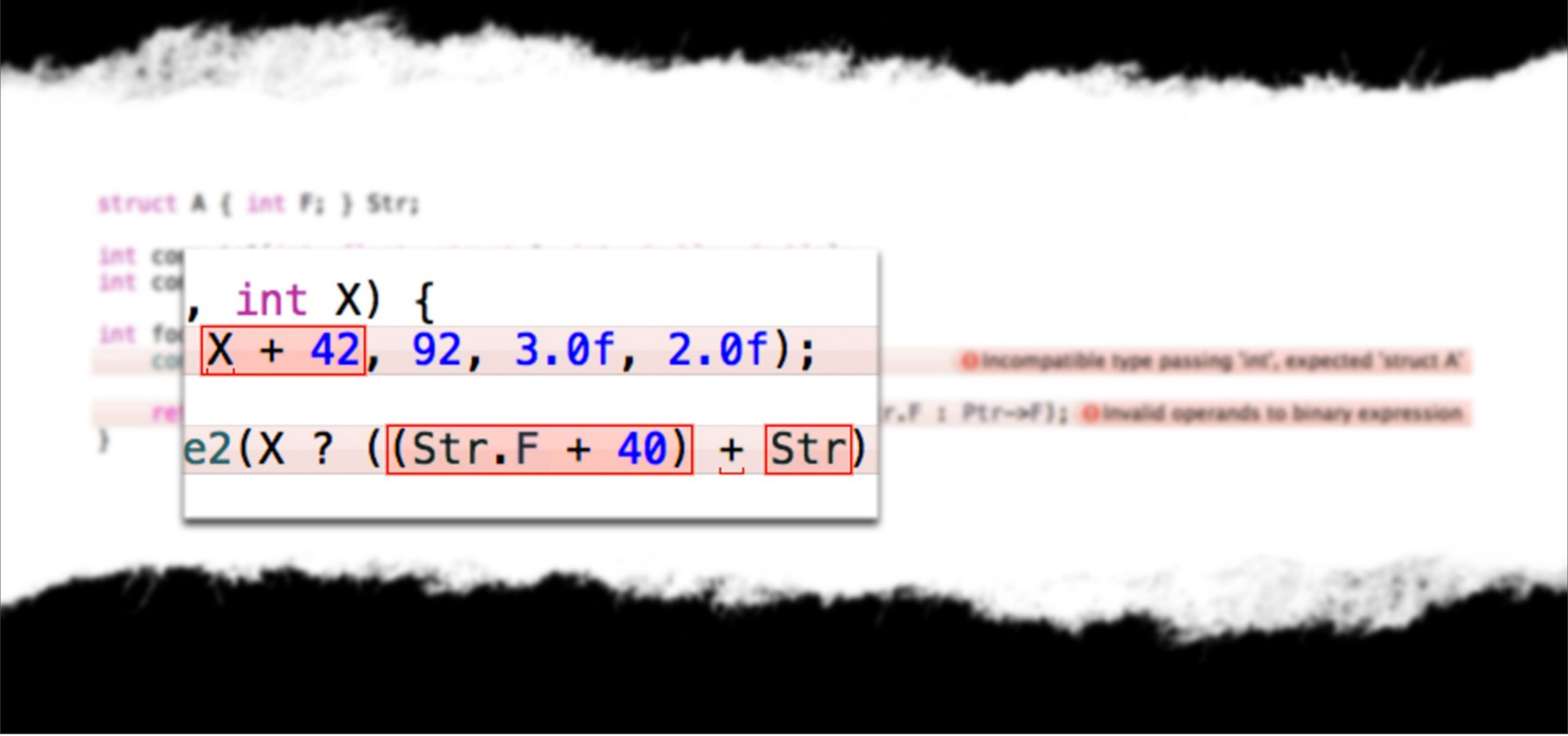
Diagnostics

Clang error messages in Xcode

```
struct A { int F; } Str;

int compute1(int, float, struct A, int, double, double);
int compute2(int);

int foo(struct A *Ptr, int X) {
    compute1(X, 4.0f, X + 42, 92, 3.0f, 2.0f);          Incompatible type passing 'int', expected 'struct A'

    return X + compute2(X ? ((Str.F + 40) + Str) / 42 + Str.F : Ptr->F);  Invalid operands to binary expression
}
```

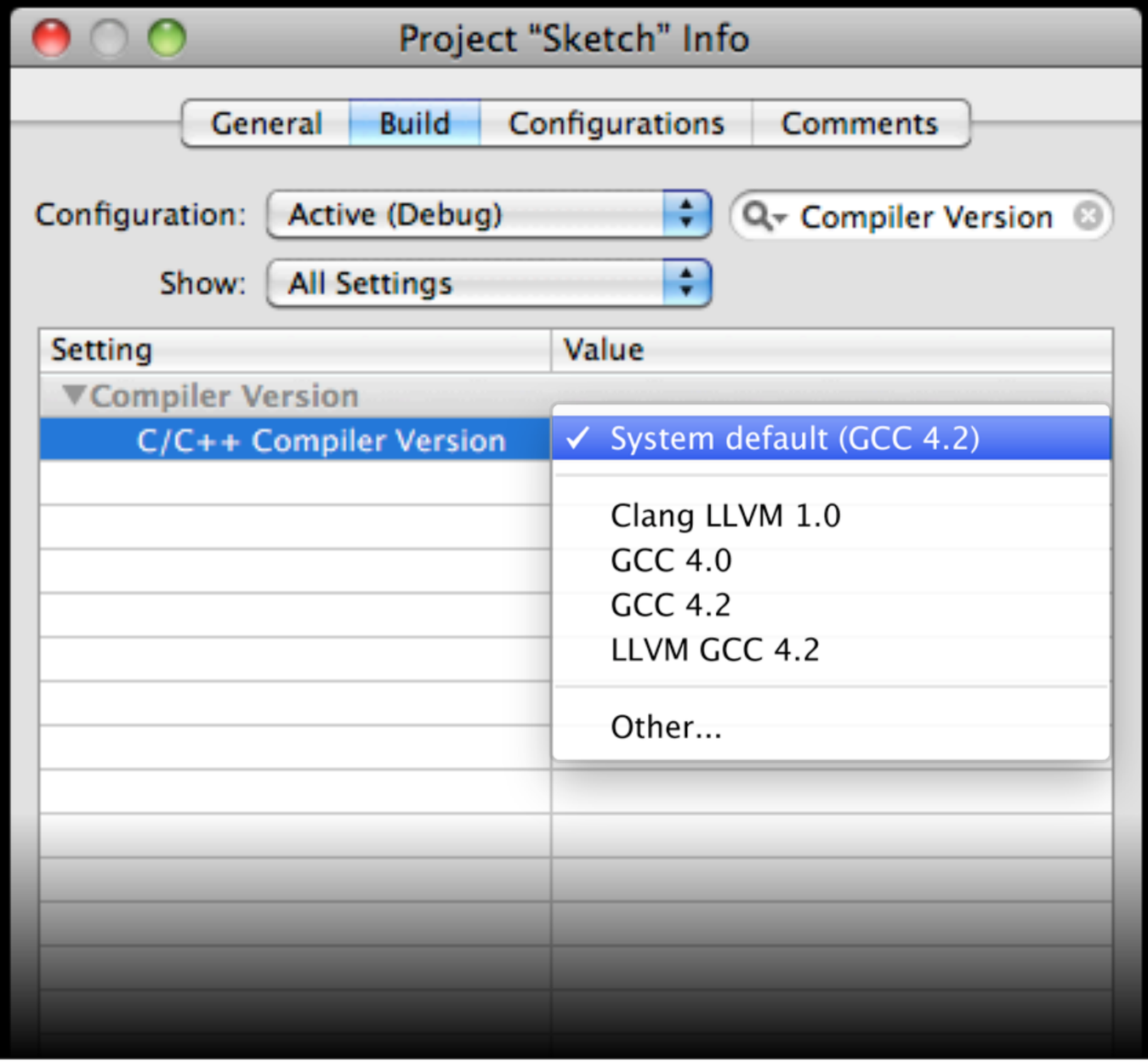
Diagnostics

Clang error messages in Xcode

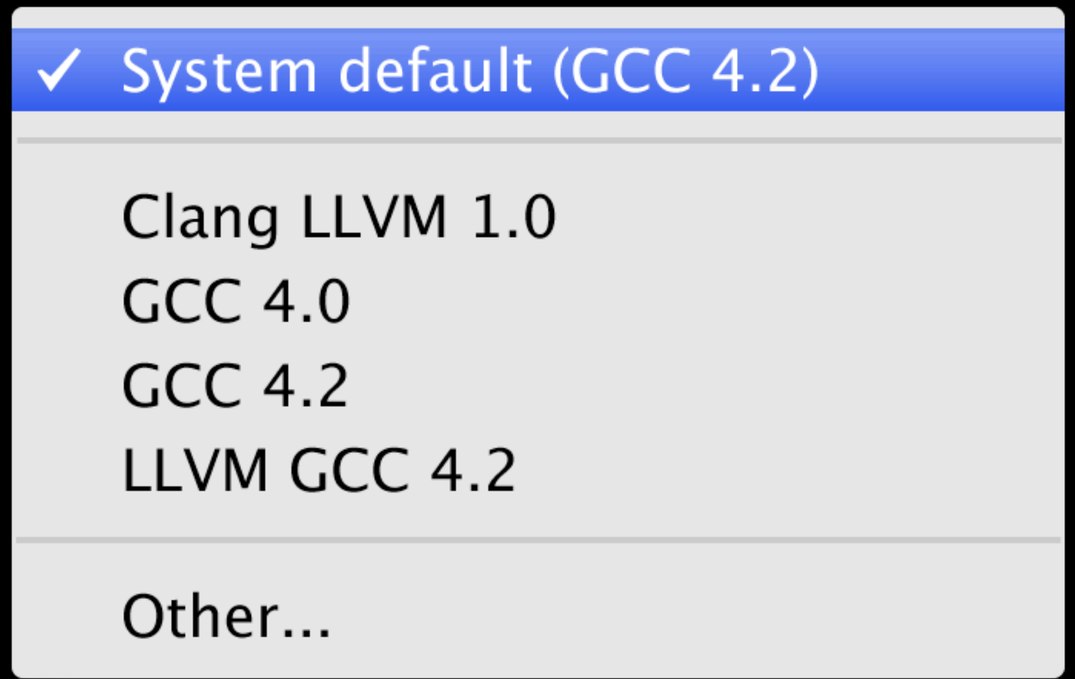


LLVM and Clang Score Card

One Click in Xcode

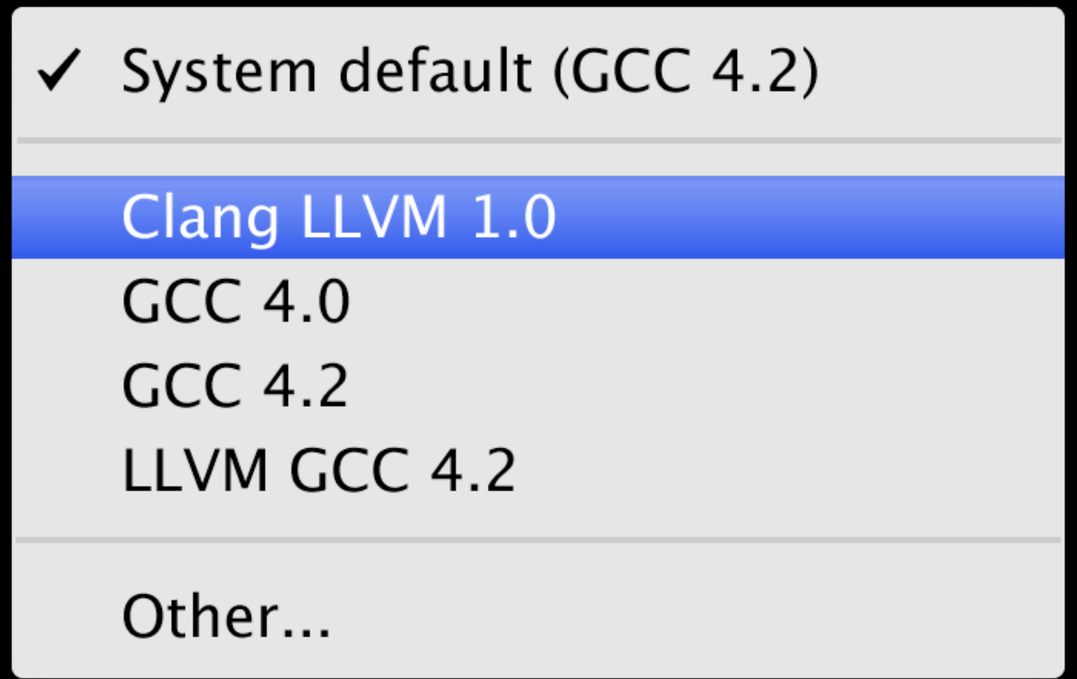


Default



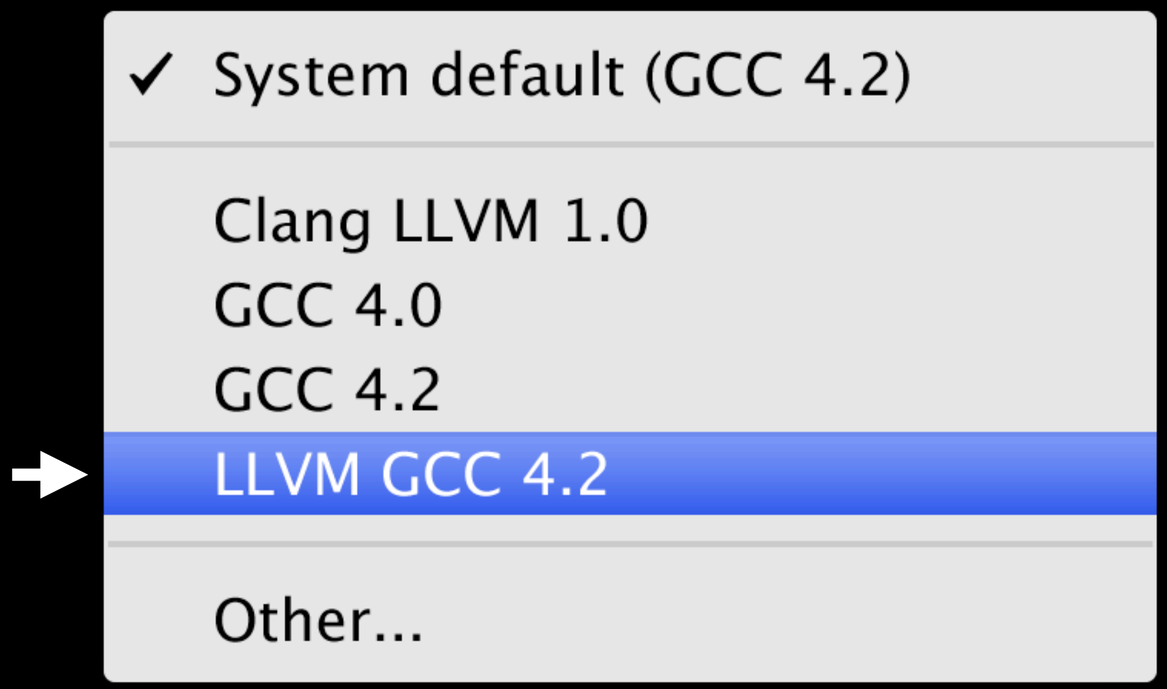
LLVM and GCC compilers fully binary compatible

Best experience



Fastest builds
Fastest code

Most compatible



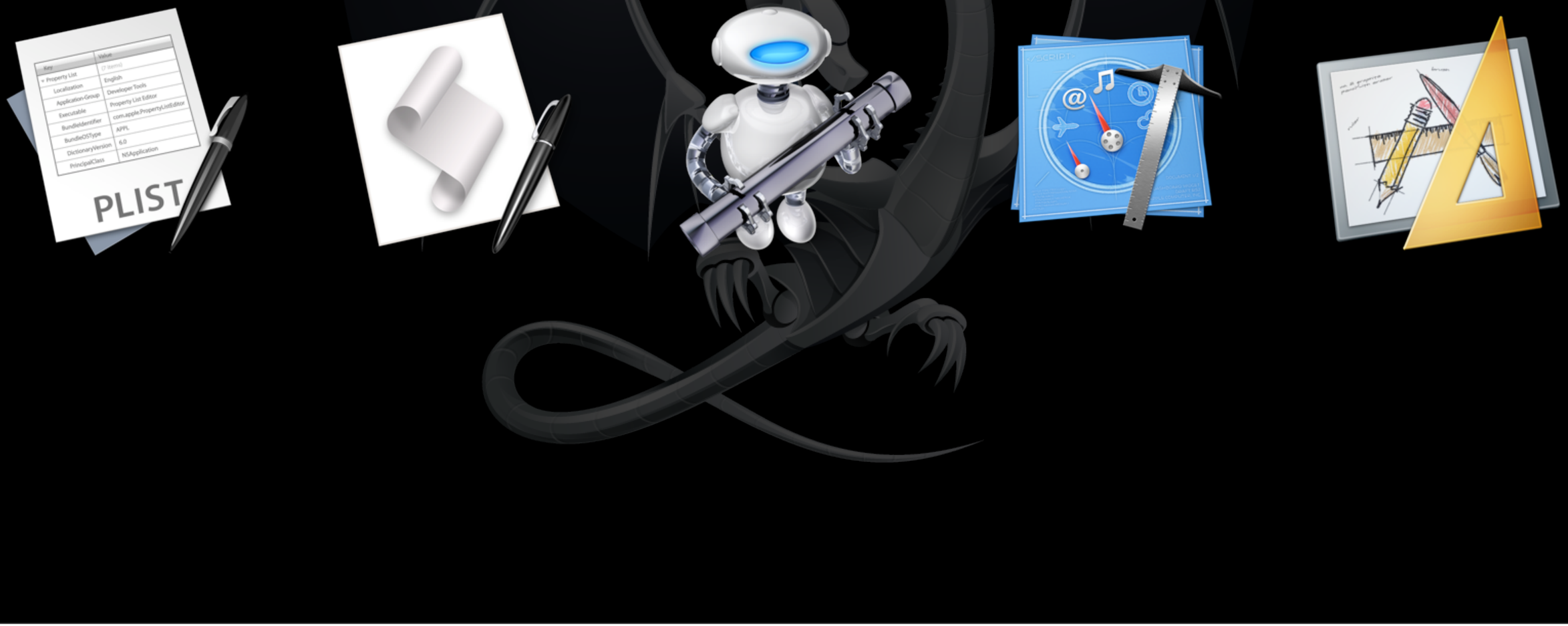
Source code compatible

Fastest code

LLVM and Clang

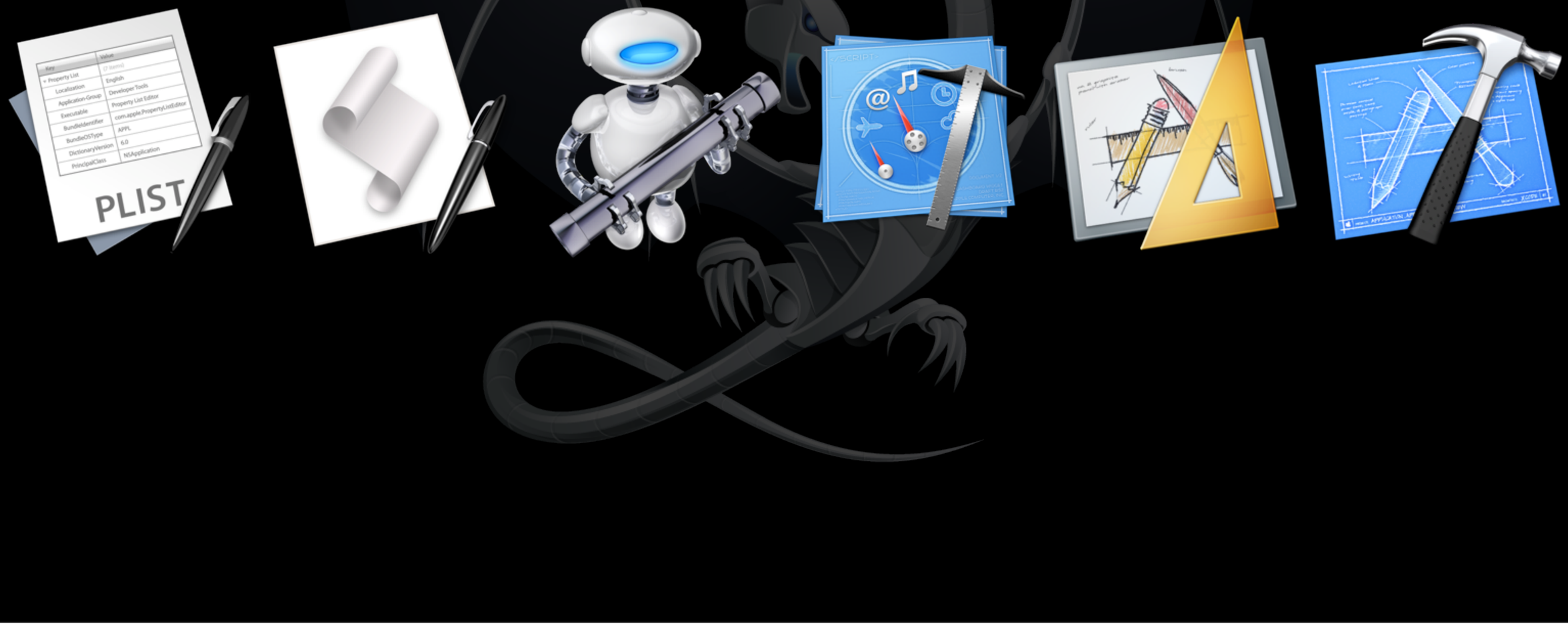
Applications Built with Clang-LLVM

Shipping in Snow Leopard



Applications Built with Clang-LLVM

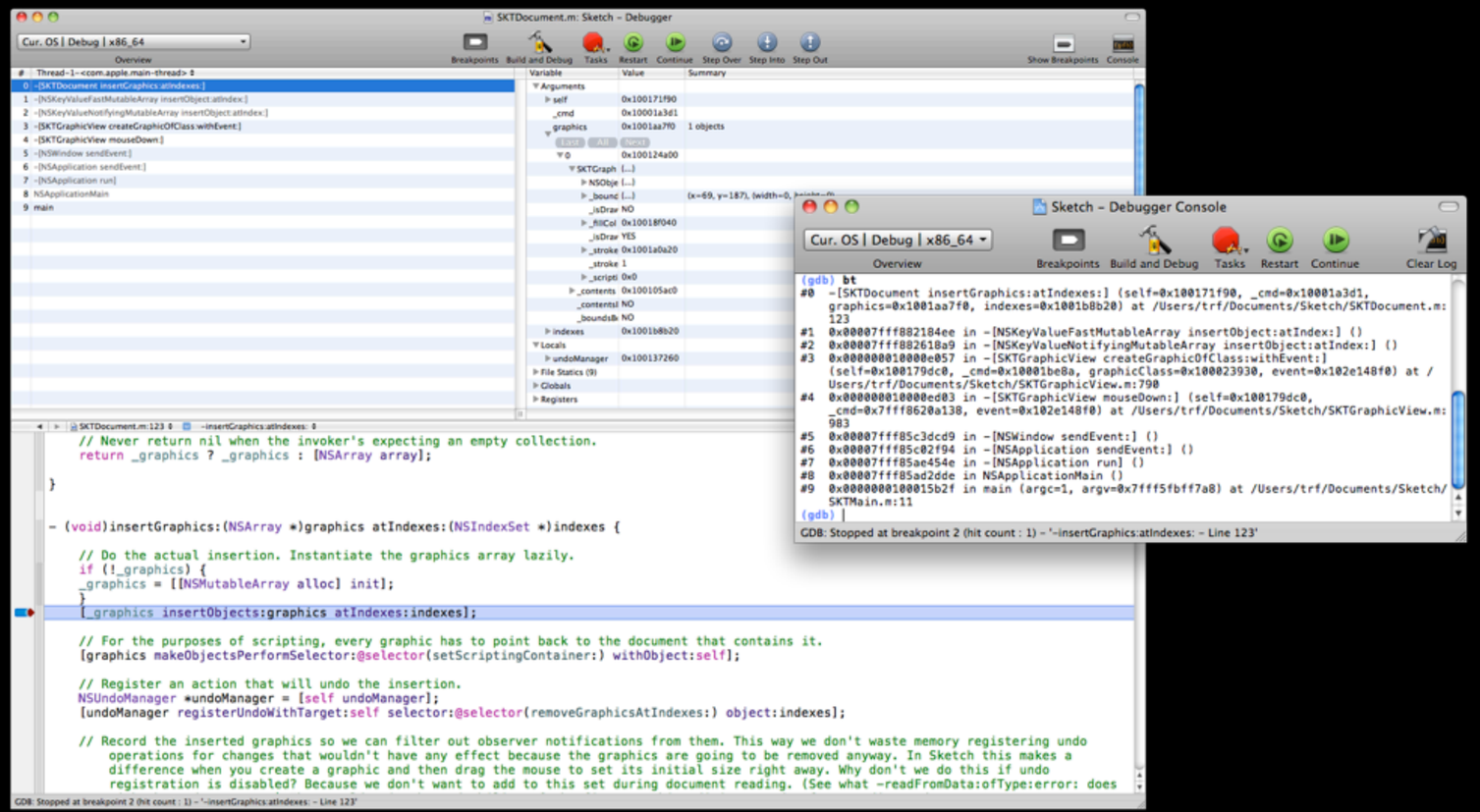
Shipping in Snow Leopard



Debugger

Integrated into Xcode

- Editor-centric workflow for most tasks
- Debugger and Console windows for more advanced tasks



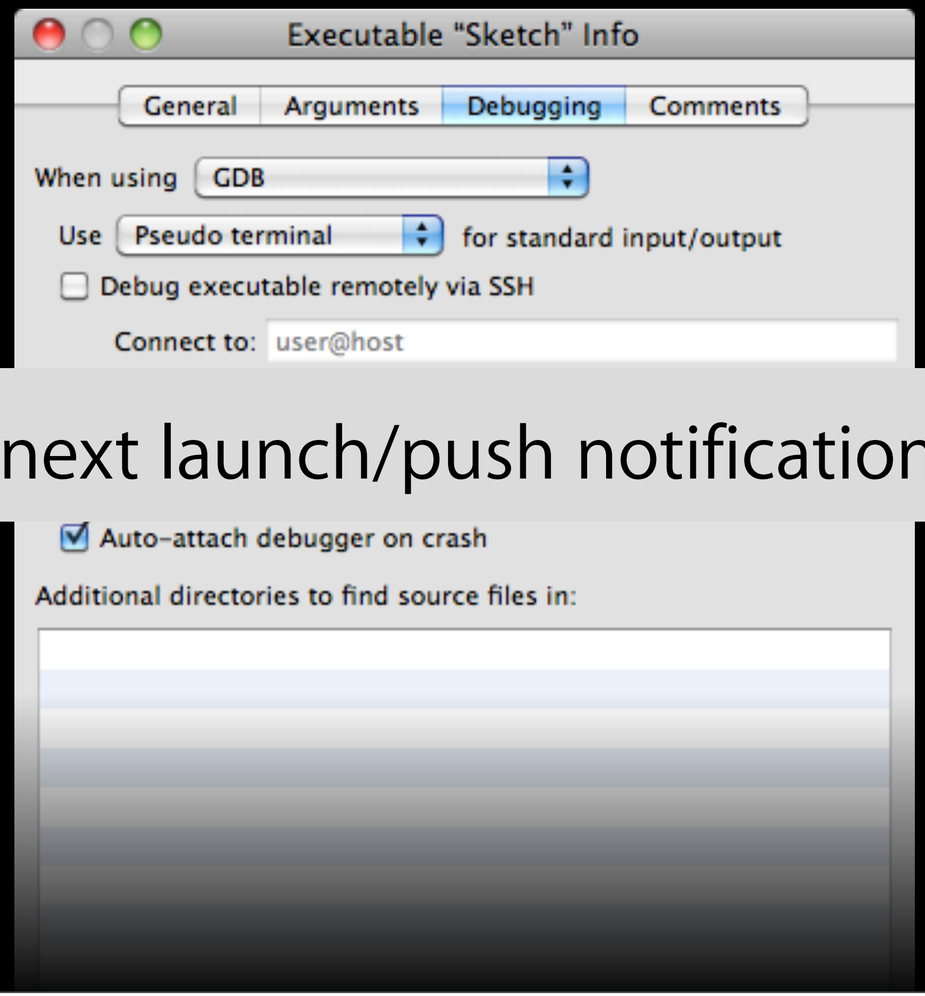
Debugger

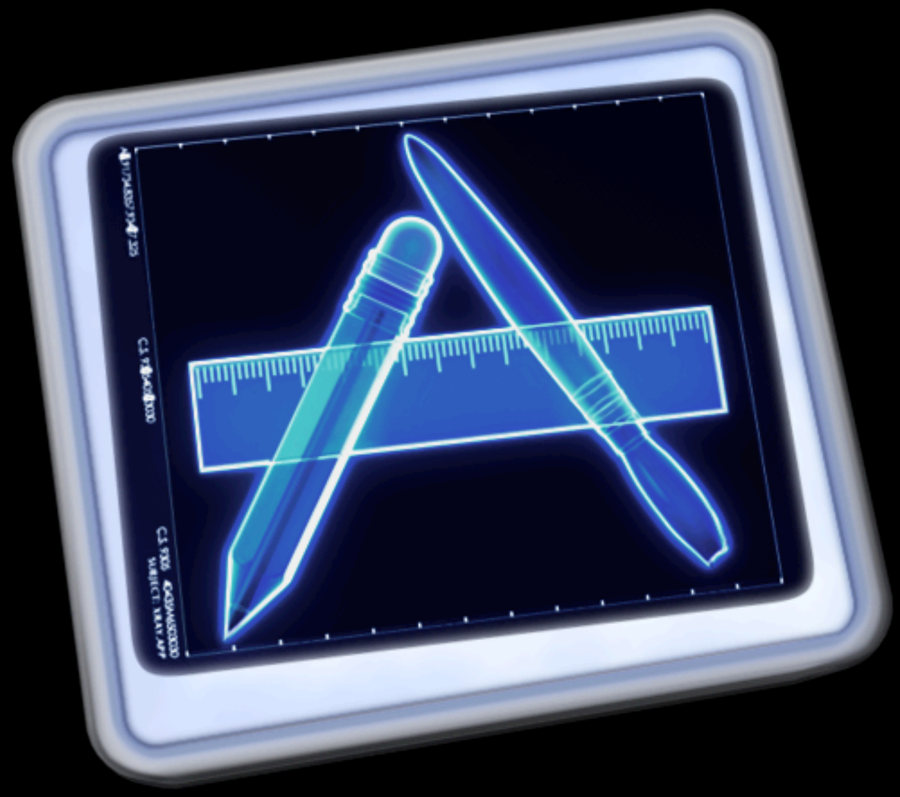
Remote debugger

Debugger

New: On-demand processes

- Suspend process launch
- Debug different on-demand code paths
  - iPhone push notifications
  - Spotlight importers
  - Daemons
- Activate in executable inspector

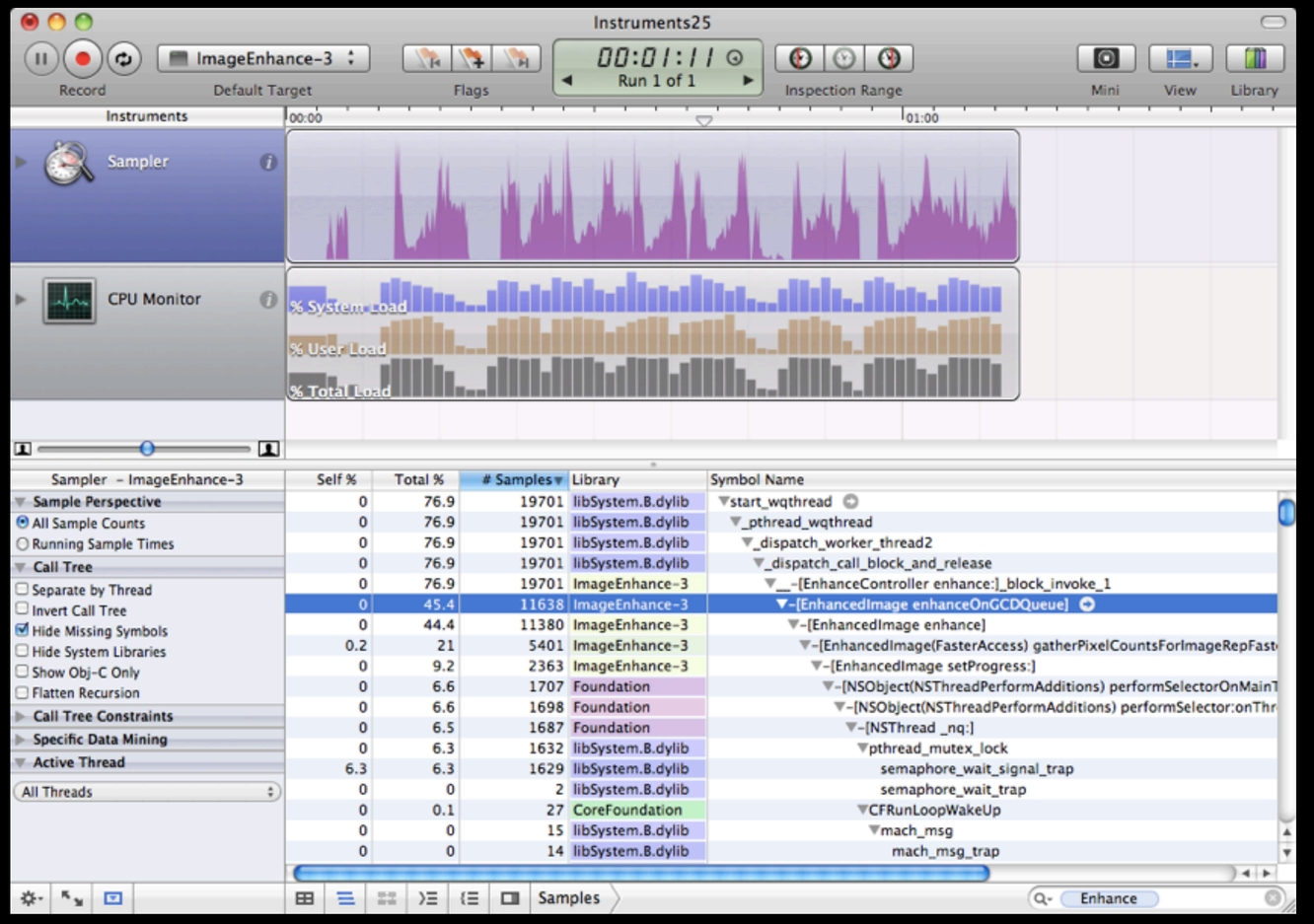




Instruments

Correlating data sources

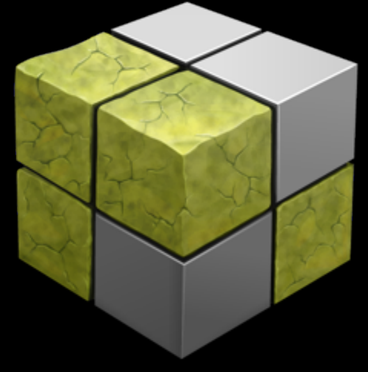
- Garage Band like user interface
- Visually correlate data points
- Reveal hidden interactions
- No recompilation required



Instruments

Types of data collectors

- Many refinements to instruments from previous releases
  - CPU
  - Graphics
  - Memory
  - Network
  - File system
- New instruments for Snow Leopard
  - GCD
  - Threads
  - Instant Off
  - Zombies

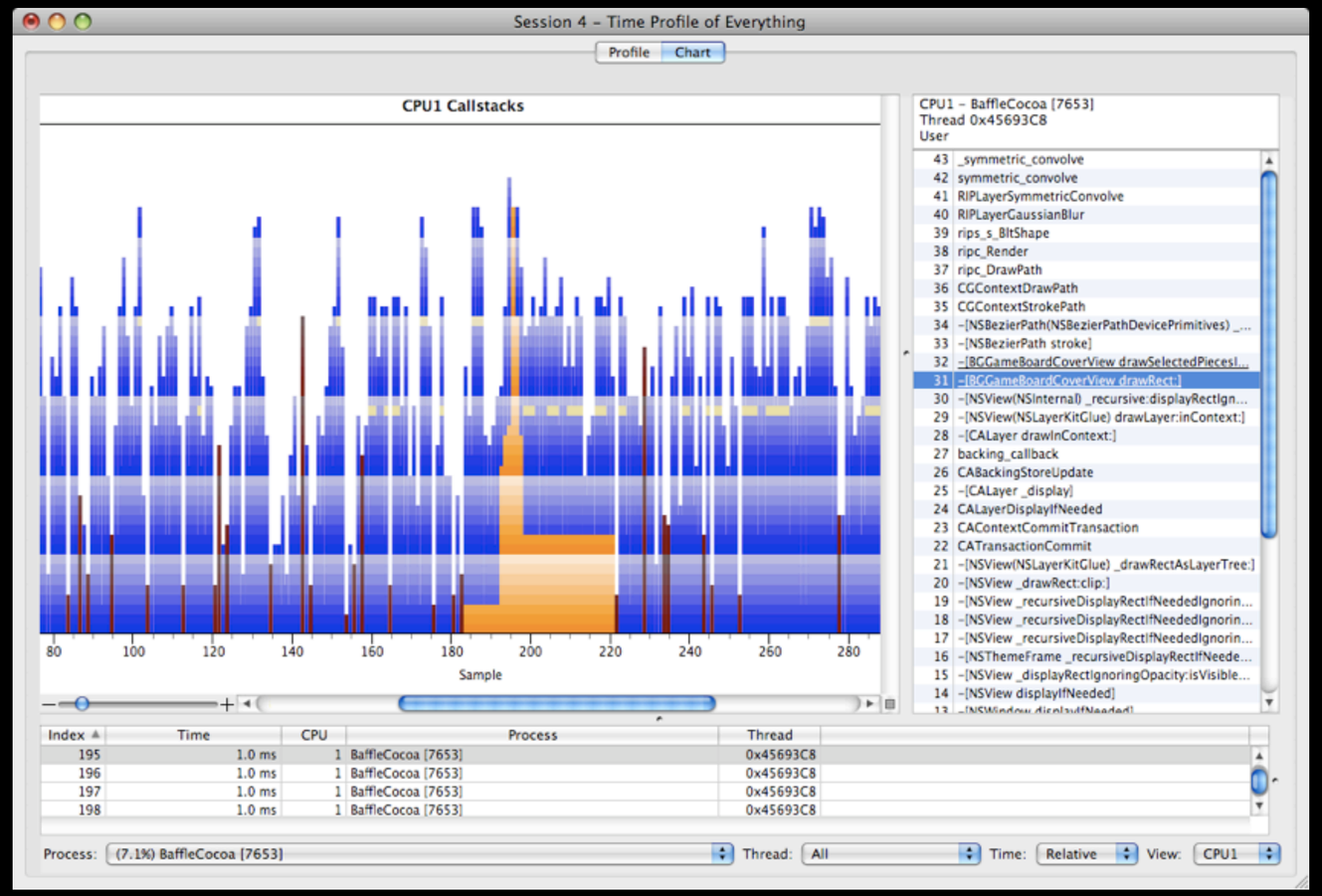


0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1
0 1 0 1 0 1

Shark

Powerful profiling

- Very low overhead
- Fine grained sampling
- Application and full system traces
- No recompilation required



Combine Strengths of Instruments and Shark

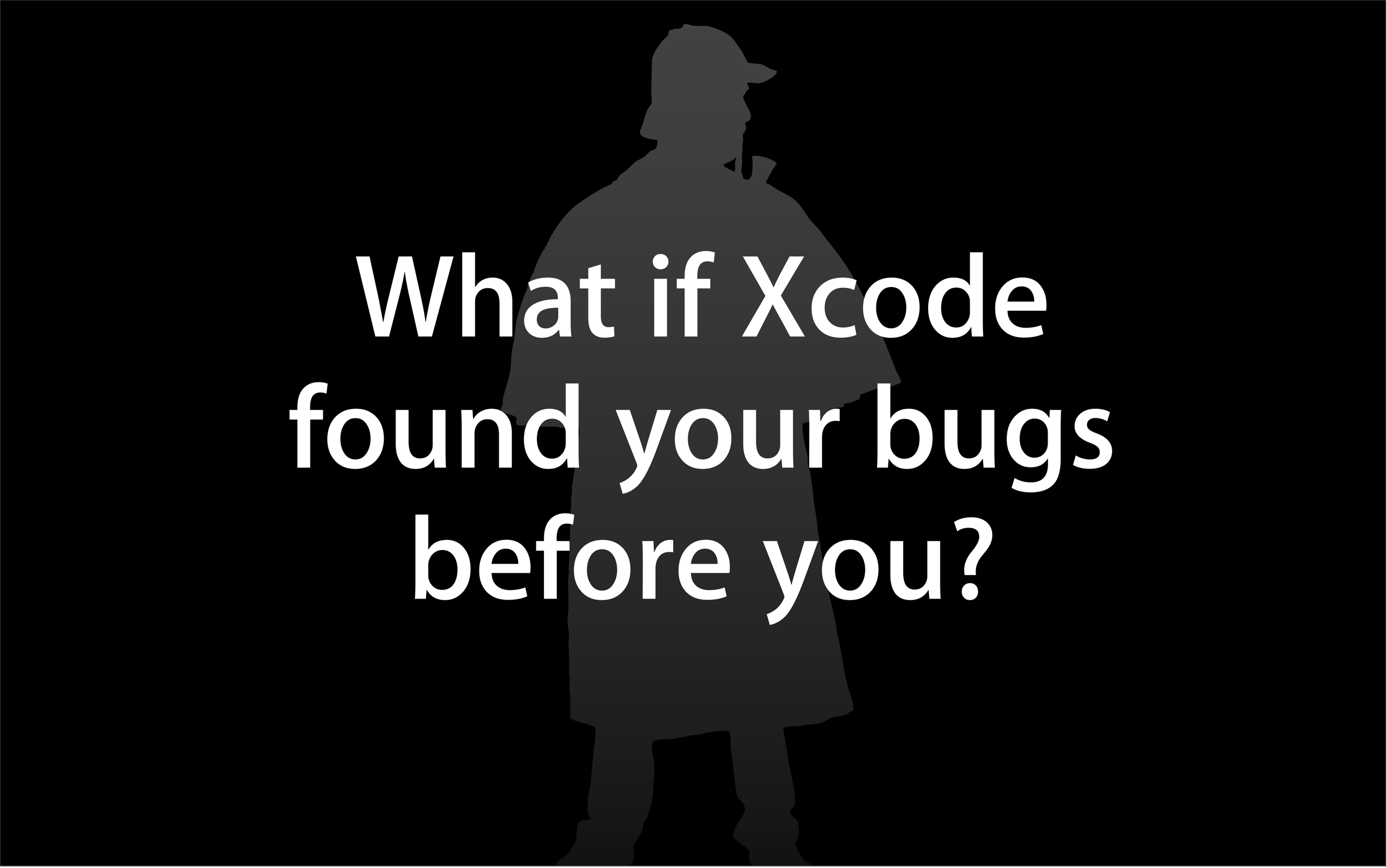
Best of both worlds

- Advanced analysis features
  - Annotated source code viewing
  - Call tree charging
- New Time Profiler instrument
  - Lowest overhead ever
  - Shark technology to collect data
  - Instruments application to analyze data

Demo

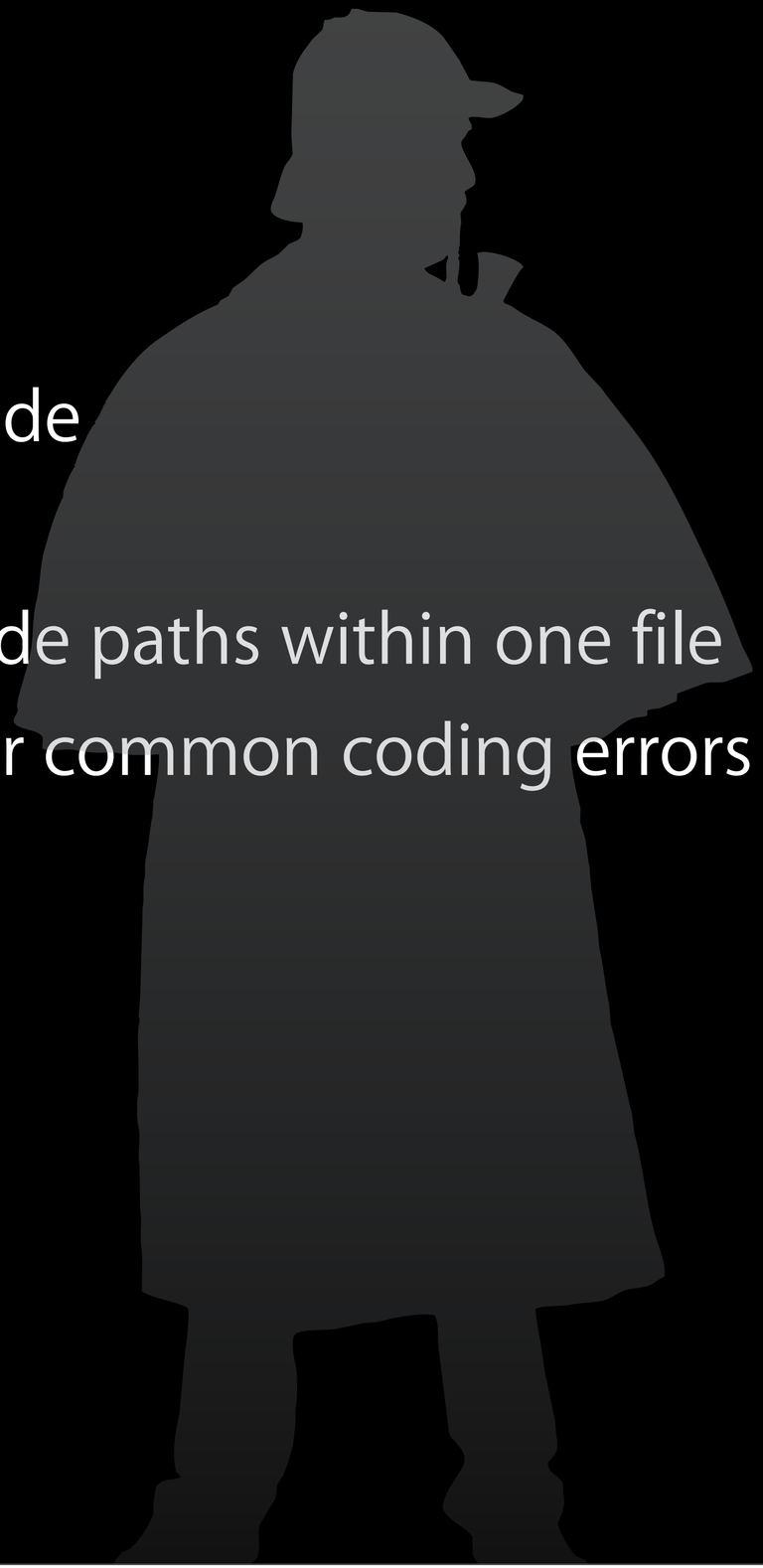
Analyzing a Mac application with Instruments

Steve Lewallen
Sr. Manager, Performance Tools


What if Xcode
found your bugs
before you?

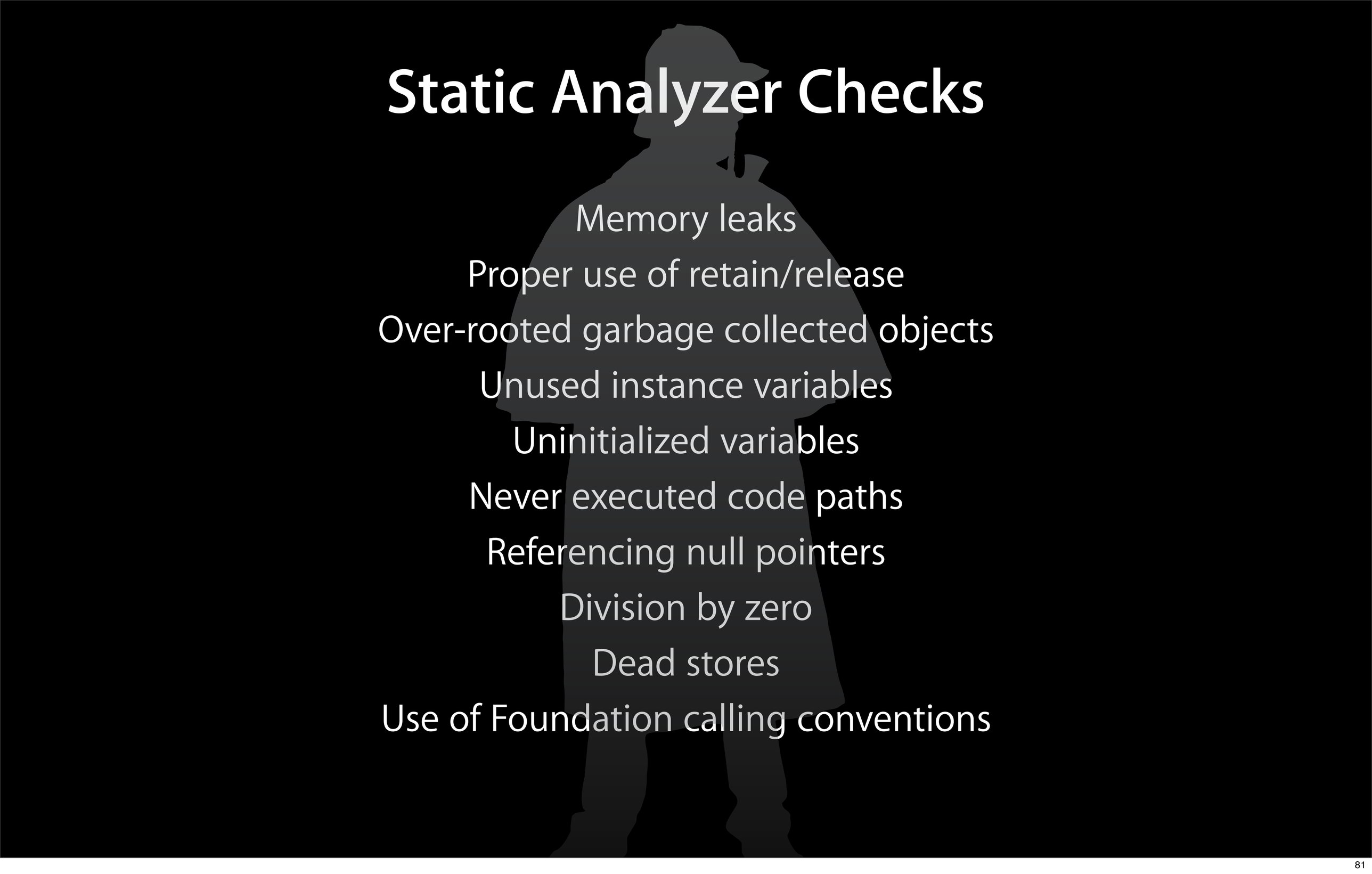
Static Analyzer

Based on Clang



- Dive deep into your code
- Objective-C and C
- Analyze all possible code paths within one file
- Automatically check for common coding errors

Static Analyzer Checks



Memory leaks

Proper use of retain/release

Over-rooted garbage collected objects

Unused instance variables

Uninitialized variables

Never executed code paths

Referencing null pointers

Division by zero

Dead stores

Use of Foundation calling conventions

Mac and
iPhone

Demo

Static Analyzer in Xcode

Todd Fernandez
Sr. Manager, IDEs

1000+ Bugs
Found and Fixed

WWDC09